

UNITED STATES AIR FORCE
SUMMER RESEARCH PROGRAM -- 1995
SUMMER RESEARCH EXTENSION PROGRAM FINAL REPORTS

VOLUME 4B
WRIGHT LABORATORY

RESEARCH & DEVELOPMENT LABORATORIES
5800 Uplander Way
Culver City, CA 90230-6608

Program Director, RDL
Gary Moore

Program Manager, AFOSR
Major David Hart

Program Manager, RDL
Scott Licoscos

Program Administrator, RDL
Gwendolyn Smith

Submitted to:

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
Bolling Air Force Base
Washington, D.C.
May 1996

20010319 025

AQM01-06-0911

REPORT DOCUMENTATION PAGE

88

AFRL-SR-BL-TR-00-

completing and reviewing
retractate for information

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information to Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE

May, 1996

0700

4. TITLE AND SUBTITLE

1995 Summer Research Program (SRP), Summer Research Extension Program (SREP), Final Report, Volume 4B, Wright Laboratory

5. FUNDING NUMBERS

F49620-93-C-0063

6. AUTHOR(S)

Gary Moore

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Research & Development Laboratories (RDL)
5800 Uplander Way
Culver City, CA 90230-6608

8. PERFORMING ORGANIZATION
REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Office of Scientific Research (AFOSR)
801 N. Randolph St.
Arlington, VA 22203-1977

10. SPONSORING/MONITORING
AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The United States Air Force Summer Research Program (SRP) is designed to introduce university, college, and technical institute faculty members to Air Force research. This is accomplished by the faculty members, graduate students, and high school students being selected on a nationally advertised competitive basis during the summer intersession period to perform research at Air Force Research Laboratory (AFRL) Technical Directorates and Air Force Air Logistics Centers (ALC). AFOSR also offers its research associates (faculty only) an opportunity, under the Summer Research Extension Program (SREP), to continue their AFOSR-sponsored research at their home institutions through the award of research grants. This volume consists of the SREP program background, management information, statistics, a listing of the participants, and the technical report for each participant of the SREP working at the AF Wright Laboratory.

14. SUBJECT TERMS

Air Force Research, Air Force, Engineering, Laboratories, Reports, Summer, Universities, Faculty, Graduate Student, High School Student

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION
OF REPORT

Unclassified

18. SECURITY CLASSIFICATION
OF THIS PAGE

Unclassified

19. SECURITY CLASSIFICATION
OF ABSTRACT

Unclassified

20. LIMITATION OF
ABSTRACT

UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to *stay within the lines* to meet *optical scanning requirements*.

Block 1. Agency Use Only (Leave blank).

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C - Contract
G - Grant
PE - Program Element

PR - Project
TA - Task
WU - Work Unit
Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. //f known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with....; Trans. of....; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."

DOE - See authorities.

NASA - See Handbook NHB 2200.2.

NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - Leave blank.

DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.

Leave blank.

NASA - Leave blank.

NTIS -

Block 13. Abstract. Include a brief (*Maximum 200 words*) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (*NTIS only*).

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

PREFACE

This volume is part of a five-volume set that summarizes the research of participants in the 1995 AFOSR Summer Research Extension Program (SREP). The current volume, Volume 1 of 5, presents the final reports of SREP participants at Armstrong Laboratory, Phillips Laboratory, Rome Laboratory, Wright Laboratory, Arnold Engineering Development Center, Frank J. Seiler Research Laboratory, and Wilford Hall Medical Center.

Reports presented in this volume are arranged alphabetically by author and are numbered consecutively -- e.g., 1-1, 1-2, 1-3; 2-1, 2-2, 2-3, with each series of reports preceded by a management summary. Reports in the five-volume set are organized as follows:

VOLUME	TITLE
1A	Armstrong Laboratory (part one)
1B	Armstrong Laboratory (part two)
2	Phillips Laboratory
3	Rome Laboratory
4A	Wright Laboratory (part one)
4B	Wright Laboratory (part two)
5	Arnold Engineering Development Center Frank J. Seiler Research Laboratory Wilford Hall Medical Center

1995 SREP FINAL REPORTS

Armstrong Laboratory

VOLUME 1

Report #	Report Title Author's University	Report Author
1	Determination of the Redox Capacity of Soil Sediment and Prediction of Pollutant University of Georgia, Athens, GA	Dr. James Anderson Analytical Chemistry AL/EQ
2	Finite Element Modeling of the Human Neck and Its Validation for the ATB Villanova University, Villanova, PA	Dr. Hashem Ashrafiou Mechanical Engineering AL/CF
3	An Examination of the Validity of the Experimental Air Force ASVAB Composites Tulane University, New Orleans, LA	Dr. Michael Burke Psychology AL/HR
4	Fuel Identification by Neural Networks Analysis of the Response of Vapor Sensitive Sensors Arrays Edinboro University of Pennsylvania, Edinboro, PA	Dr. Paul Edwards Chemistry AL/EQ
5	A Comparison of Multistep vs Singlestep Arrhenius Integral Models for Describing Laser Induced Thermal Damage Florida International University, Miami, FL	Dr. Bernard Gerstman Physics AL/OE
6	Effects of Mental Workload and Electronic Support on Negotiation Performance University of Dayton, Dayton, OH	Dr. Kenneth Graetz Psychology AL/HR
7	Regression to the Mean in Half Life Studies University of Maine, Orono, ME	Dr. Pushpa Gupta Mathematics & Statistics AL/AO
8	Application of the MT3D Solute Transport Model to the Made-2 Site: Calibration Florida State University, Tallahassee, FL	Dr. Manfred Koch Geophysics AL/EQ
9	Computer Calculations of Gas-Phase Reaction Rate Constants Florida State University, Tallahassee, FL	Dr. Mark Novotny Supercomp Comp. Res. I AL/EQ
10	Surface Fitting Three Dimensional Human Scan Data Ohio University, Athens, OH	Dr. Joseph Nurre Mechanical Engineering AL/CF
11	The Effects of Hyperbaric Oxygenation on Metabolism of Drugs and Other Xenobiotics University of South Carolina, Columbia, South Carolina	Dr. Edward Piepmeier Pharmaceutics AL/AO
12	Maintaining Skills After Training: The Role of Opportunity to Perform Trained Tasks on Training Effectiveness Rice University, Houston, TX	Dr. Miguel Quinones Psychology AL/HR

1995 SREP FINAL REPORTS

Armstrong Laboratory

VOLUME 1 (cont.)

Report #	Report Title Author's University	Report Author
13	Nonlinear Transcutaneous Electrical Stimulation of the Vestibular System University of Illinois Urbana-Champaign, Urbana, IL	Dr. Gary Riccio Psychology AL/CF
14	Documentation of Separating and Separated Boundary Layer Flow, For Application Texas A&M University, College Station, TX	Dr. Wayne Shebilske Psychology AL/HR
15	Tactile Feedback for Simulation of Object Shape and Textural Information in Haptic Displays Ohio State University, Columbus, OH	Dr. Janet Weisenberger Speech & Hearing AL/CF
16	Melatonin Induced Prophylactic Sleep as a Countermeasure for Sleep Deprivation Oregon Health Sciences University, Portland, OR	Mr. Rod Hughes Psychology AL/CF

1995 SREP FINAL REPORTS

Phillips Laboratory

VOLUME 2A

Report #	Report Title Author's University	Report Author
1	Investigation of the Mixed-Mode Fracture Behavior of Solid Propellants University of Houston, Houston, TX	Dr. K. Ravi-Chandar Aeronautics PL/RK
2	Performance Study of ATM-Satellite Network SUNY-Buffalo, Buffalo, NY	Dr. Nasser Ashgriz Mechanical Engineering PL/RK
3	Characterization of CMOS Circuits Using a Highly Calibrated Low-Energy X-Ray Source Embry-Riddle Aeronautical Univ., Prescott, AZ	Dr. Raymond Bellem Computer Science PL/VT
4	Neutron Diagnostics for Pulsed Plasmas of Compact Toroid-Marauder Type Stevens Institute of Tech, Hoboken, NJ	Dr. Jan Brzosko Nuclear Physics PL/WS
5	Parallel Computation of Zernike Aberration Coefficients for Optical Aberration Correction University of Houston-Victoria, Victoria, TX	Dr. Meledath Damodaran Math & Computer Science PL/LI
6	Quality Factor Evaluation of Complex Cavities University of Denver, Denver, CO	Dr. Ronald DeLyser Electrical Engineering PL/WS
7	Unidirectional Ring Lasers and Laser Gyros with Multiple Quantum Well Gain University of New Mexico, Albuquerque, NM	Dr. Jean-Claude Diels Physics PL/LI
8	A Tool for the Formation of Variable Parameter Inverse Synthetic Aperture Radar University of Nevada, Reno, NV	Dr. James Henson Electrical Engineering PL/WS
9	Radar Ambiguity Functionals Univ. of Massachusetts at Lowell, Lowell, MA	Dr. Gerald Kaiser Physics PL/GP
10	The Synthesis and Chemistry of Peroxonitrites Peroxonitrous Acid Univ. of Massachusetts at Lowell, Lowell, MA	Dr. Albert Kowalak Chemistry PL/GP
11	Temperature and Pressure Dependence of the Band Gaps and Band Offsets University of Houston, Houston, TX	Dr. Kevin Malloy Electrical Engineering PL/VT
12	Theoretical Studies of the Performance of Novel Fiber-Coupled Imaging Interferometers University of New Mexico, Albuquerque, NM	Dr. Sudhakar Prasad Physics PL/LI

1995 SREP FINAL REPORTS

Phillips Laboratory

VOLUME 2B

Report #	Author's University	Report Author
13	Static and Dynamic Graph Embedding for Parallel Programming Texas AandM Univ.-Kingsville, Kingsville, TX	Dr. Mark Purtill Mathematics PL/WS
14	Ultrafast Process and Modulation in Iodine Lasers University of New Mexico, Albuquerque, NM	Dr. W. Rudolph Physics PL/LI
15	Impedance Matching and Reflection Minimization for Transient EM Pulses Through University of New Mexico, Albuquerque, NM	Dr. Alexander Stone Mathematics and Statics PL/WS
16	Low Power Retromodular Based Optical Transceiver for Satellite Communications Utah State University, Logan, UT	Dr. Charles Swenson Electrical Engineering PL/VT
17	Improved Methods of Tilt Measurement for Extended Images in the Presence of Atmospheric Disturbances Using Optical Flow Michigan Technological Univ., Houghton, MI	Mr. John Lipp Electrical Engineering PL/LI
18	Thermoluminescence of Simple Species in Molecular Hydrogen Matrices Cal State Univ.-Northridge, Northridge, CA	Ms. Janet Petroski Chemistry PL/RK
19	Design, Fabrication, Intelligent Cure, Testing, and Flight Qualification University of Cincinnati, Cincinnati, OH	Mr. Richard Salasovich Mechanical Engineering PL/VT

1995 SREP FINAL REPORTS

Rome Laboratory

VOLUME 3

<u>Report #</u>	<u>Author's University</u>	<u>Report Author</u>
1	Performance Study of an ATM/Satellite Network Florida Atlantic University, Boca Raton, FL	Dr. Valentine Aalo Electrical Engineering RL/C3
2	Interference Excision in Spread Spectrum Communication Systems Using Time-Frequency Distributions Villanova University, Villanova, PA	Dr. Moeness Amin Electrical Engineering RL/C3
3	Designing Software by Reformulation Using KIDS Oklahoma State University, Stillwater, OK	Dr. David Benjamin Computer Science RL/C3
4	Detection Performance of Over Resolved Targets with Non-Uniform and Non-Gaussian Howard University, Washington, DC	Dr. Ajit Choudhury Engineering RL/OC
5	Computer-Aided-Design Program for Solderless Coupling Between Microstrip and stripline Structures Southern Illinois University, Carbondale, IL	Dr. Frances Harackiewicz Electrical Engineering RL/ER
6	Spanish Dialect Identification Project Colorado State University, Fort Collins, CO	Dr. Beth Losiewicz Psycholinguistics RL/IR
7	Automatic Image Registration Using Digital Terrain Elevation Data University of Maine, Orono, ME	Dr. Mohamed Musavi Engineering RL/IR
8	Infrared Images of Electromagnetic Fields University of Colorado, Colorado Springs, CO	Dr. John Norgard Engineering RL/ER
9	Femtosecond Pump-Probe Spectroscopy System SUNY Institute of Technology, Utica, NY	Dr. Dean Richardson Photonics RL/OC
10	Synthesis and Properties B-Diketonate-Modified Heterobimetallic Alkoxides Tufts University, Medford, MA	Dr. Daniel Ryder, Jr. Chemical Engineering RL/ER
11	Optoelectronic Study of Semiconductor Surfaces and Interfaces Rensselaer Polytechnic Institute, Troy, NY	Dr. Xi-Cheng Zhang Physics RL/ER

1995 SREP FINAL REPORTS

Wright Laboratory

VOLUME 4A

<u>Report #</u>	<u>Author's University</u>	<u>Report Author</u>
1	An Investigation of the Heating and Temperature Distribution in Electrically Excited Foils Auburn University, Auburn, AL	Dr. Michael Baginski Electrical Engineering WL/MN
2	Micromechanics of Creep in Metals and Ceramics at High Temperature Wayne State University, Detroit, MI	Dr. Victor Berdichevsky Aerospace Engineering WL/FI
3	Development of a Fluorescence-Based Chemical Sensor for Simultaneous Oxygen Quantitation and Temp. Measurement Columbus College, Columbus, GA	Dr. Steven Buckner Chemistry WL/PO
4	Development of High-Performance Active Dynamometer Sys. for Machines and Drive Clarkson University, Potsdam, NY	Dr. James Carroll Electrical Engineering WL/PO
5	SOLVING $z(t)=\ln[A\cos(\omega_1 t)+B\cos(\omega_2 t)+C]$ Transylvania University, Lexington, KY	Dr. David Choate Mathematics WL/AA
6	Synthesis, Processing and Characterization of Nonlinear Optical Polymer Thin Films University of Cincinnati, Cincinnati, OH	Dr. Stephen Clarson Matls Science& Engineering WL/ML
7	An Investigation of Planning and Scheduling Algorithms for Sensor Management Embry-Riddle Aeronautical University, Prescott, AZ	Dr. Milton Cone Comp. Science & Engineering WL/AA
8	A Study to Determine Wave Gun Firing Cycles for High Performance Model Launches Louisiana State University, Baton Rouge, LA	Dr. Robert Courter Mechanical Engineering WL/MN
9	Characterization of Electro-Optic Polymers University of Dayton, Dayton, OH	Dr. Vincent Dominic Electro Optics Program WL/ML
10	A Methodology for Affordability in the Design Process Clemson Univeristy, Clemson, SC	Dr. Georges Fadel Mechanical Engineering WL/MT
11	Data Reduction and Analysis for Laser Doppler Velocimetry North Carolina State University, Raleigh, NC	Dr. Richard Gould Mechanical Engineering WL/PO

1995 SREP FINAL REPORTS

Wright Laboratory

VOLUME 4A (cont.)

<u>Report #</u>	<u>Author's University</u>	<u>Report Author</u>
12	Hyperspectral Target Identification Using Bomen Spectrometer Data University of Dayton, Dayton, OH	Dr. Russell Hardie Electrical Engineering WL/AA
13	Robust Fault Detection and Classification Auburn University, Auburn, AL	Dr. Alan Hodel Electrical Engineering WL/MN
14	Multidimensional Algorithm Development and Analysis Mississippi State University, Mississippi State University, MS	Dr. Jonathan Janus Aerospace Engineering WL/MN
15	Characterization of Interfaces in Metal-Matrix Composites Michigan State University, East Lansing, MI	Dr. Iwona Jasiuk Materials Science WL/ML
16	TSI Mitigation: A Mountaintop Database Study Lafayette College, Easton, PA	Dr. Ismail Jouny Electrical Engineering WL/AA
17	Comparative Study and Performance Analysis of High Resolution SAR Imaging Techniques University of Florida, Gainesville, FL	Dr. Jian Li Electrical Engineering WL/AA

1995 SREP FINAL REPORTS

Wright Laboratory

VOLUME 4B

Report #	Author's University	Report Author
18	Prediction of Missile Trajectory University of Missouri-Columbia, Columbia, MO	Dr. Chun-Shin Lin Electrical Engineering WL/FI
19	Three Dimensional Deformation Comparison Between Bias and Radial Aircraft Tires Cleveland State University, Cleveland, OH	Dr. Paul Lin Mechanical Engineering WL/FI
20	Investigation of AlGaAs/GaAs Heterojunctin Bipolar Transistor Reliability Based University of Central Florida, Orlando, FL	Dr. Juin Liou Electrical Engineering WL/EL
21	Thermophysical Invariants From LWIR Imagery for ATR University of Virginia, Charlottesville, VA	Dr. Nagaraj Nandhakumar Electrical Engineering WL/AA
22	Effect of Electromagnetic Environment on Array Signal Processing University of Dayton, Dayton, OH	Dr. Krishna Pasala Electrical Engineering WL/AA
23	Functional Decomposition of Binary, Multiple-Valued, and Fuzzy Logic Portland State University, Portland, OR	Dr. Marek Perkowski Electrical Engineering WL/AA
24	Superresolution of Passive Millimeter-Wave Imaging Auburn University, Auburn, AL	Dr. Stanley Reeves Electrical Engineering WL/MN
25	Development of a Penetrator Optimizer University of Alabama, Tuscaloosa, AL	Dr. William Rule Engineering Science WL/MN
26	Heat Transfer for Turbine Blade Film Cooling with Free Stream Turbulence-Measurements and Predictions University of Dayton, Dayton, OH	Dr. John Schauer Mech. & Aerosp. Engineering WL/FI
27	Neural Network Identification and Control in Metal Forging University of Florida, Gainesville, FL	Dr. Carla Schwartz Electrical Engineering WL/FI
28	Documentation of Separating and Separated Boundary Layer Flow, for Application University of Minnesota, Minneapolis, MN	Dr. Terrence Simon Mechanical Engineering WL/PO
29	Transmission Electron Microscopy of Semiconductor Heterojunctions Carnegie Melon University, Pittsburgh, PA	Dr. Marek Skowronski Matls Science & Engineering WL/EL

1995 SREP FINAL REPORTS

Wright Laboratory

VOLUME 4B (cont.)

<u>Report #</u>	<u>Author's University</u>	<u>Report Author</u>
30	Parser in SWI-PROLOG Wright State University, Dayton, OH	Dr. K. Thirunarayan Computer Science WL/EL
31	Development of Qualitative Process Control Discovery Systems for Polymer Composite and Biological Materials University of California, Los Angeles, CA	Dr. Robert Trelease Anatomy & Cell Biology WL/ML
32	Improved Algorithm Development of Massively Parallel Epic Hydrocode in Cray T3D Massively Parallel Computer Florida Atlantic University, Boca Raton, FL	Dr. Chi-Tay Tsai Engineering Mechanics WL/MN
33	The Characterization of the Mechanical Properties of Materials in a Biaxial Stress Environment University of Kentucky, Lexington, KY	Dr. John Lewis Materials Science Engineering WL/MN

1995 SREP FINAL REPORTS

VOLUME 5

<u>Report #</u>	<u>Author's University</u>	<u>Report Author</u>
Arnold Engineering Development Center		
1	Plant-Wide Preventive Maintenance and Monitoring Vanderbilt University	Mr. Theodore Bapty Electrical Engineering AEDC
Frank J. Seiler Research Laboratory		
1	Block Copolymers at Inorganic Solid Surfaces Colorado School of Mines, Golden, CO	Dr. John Dorgan Chemical Engineering FJSRL
2	Non-Linear Optical Properties of Polyacetylenes and Related Barry University, Miami, FL	Dr. M. A. Jungbauer Chemistry FJSRL
3	Studies of Second Harmonic Generation in Glass Waveguides Allegheny College, Meadville, PA	Dr. David Statman Physics FJSRL
Wilford Hall Medical Center		
1	Biochemical & Cell Physiological Aspects of Hyperthermia University of Miami, Coral Gables, FL	Dr. W. Drost-Hansen Chemistry WHMC

1995 SUMMER RESEARCH EXTENSION PROGRAM (SREP) MANAGEMENT REPORT

1.0 BACKGROUND

Under the provisions of Air Force Office of Scientific Research (AFOSR) contract F49620-90-C-0076, September 1990, Research & Development Laboratories (RDL), an 8(a) contractor in Culver City, CA, manages AFOSR's Summer Research Program. This report is issued in partial fulfillment of that contract (CLIN 0003AC).

The Summer Research Extension Program (SREP) is one of four programs AFOSR manages under the Summer Research Program. The Summer Faculty Research Program (SFRP) and the Graduate Student Research Program (GSRP) place college-level research associates in Air Force research laboratories around the United States for 8 to 12 weeks of research with Air Force scientists. The High School Apprenticeship Program (HSAP) is the fourth element of the Summer Research Program, allowing promising mathematics and science students to spend two months of their summer vacations working at Air Force laboratories within commuting distance from their homes.

SFRP associates and exceptional GSRP associates are encouraged, at the end of their summer tours, to write proposals to extend their summer research during the following calendar year at their home institutions. AFOSR provides funds adequate to pay for SREP subcontracts. In addition, AFOSR has traditionally provided further funding, when available, to pay for additional SREP proposals, including those submitted by associates from Historically Black Colleges and Universities (HBCUs) and Minority Institutions (MIs). Finally, laboratories may transfer internal funds to AFOSR to fund additional SREPs. Ultimately the laboratories inform RDL of their SREP choices, RDL gets AFOSR approval, and RDL forwards a subcontract to the institution where the SREP associate is employed. The subcontract (see Appendix 1 for a sample) cites the SREP associate as the principal investigator and requires submission of a report at the end of the subcontract period.

Institutions are encouraged to share costs of the SREP research, and many do so. The most common cost-sharing arrangement is reduction in the overhead, fringes, or administrative charges institutions would normally add on to the principal investigator's or research associate's labor. Some institutions also provide other support (e.g., computer run time, administrative assistance, facilities and equipment or research assistants) at reduced or no cost.

When RDL receives the signed subcontract, we fund the effort initially by providing 90% of the subcontract amount to the institution (normally \$18,000 for a \$20,000 SREP). When we receive the end-of-research report, we evaluate it administratively and send a copy to the laboratory for a technical evaluation. When the laboratory notifies us the SREP report is acceptable, we release the remaining funds to the institution.

2.0 THE 1995 SREP PROGRAM

SELECTION DATA: A total of 719 faculty members (SFRP Associates) and 286 graduate students (GSRP associates) applied to participate in the 1994 Summer Research Program. From these applicants 185 SFRPs and 121 GSRPs were selected. The education level of those selected was as follows:

1994 SRP Associates, by Degree			
SFRP		GSRP	
PHD	MS	MS	BS
179	6	52	69

Of the participants in the 1994 Summer Research Program 90 percent of SFRPs and 25 percent of GSRPs submitted proposals for the SREP. Ninety proposals from SFRPs and ten from GSRPs were selected for funding, which equates to a selection rate of 54% of the SFRP proposals and of 34% for GSRP proposals.

1995 SREP: Proposals Submitted vs. Proposals Selected			
	Summer 1994 Participants	Submitted SREP Proposals	SREPs Funded
SFRP	185	167	90
GSRP	121	29	10
TOTAL	306	196	100

The funding was provided as follows:

Contractual slots funded by AFOSR	75
Laboratory funded	14
Additional funding from AFOSR	<u>11</u>
Total	100

Six HBCU/MI associates from the 1994 summer program submitted SREP proposals; six were selected (none were lab-funded; all were funded by additional AFOSR funds).

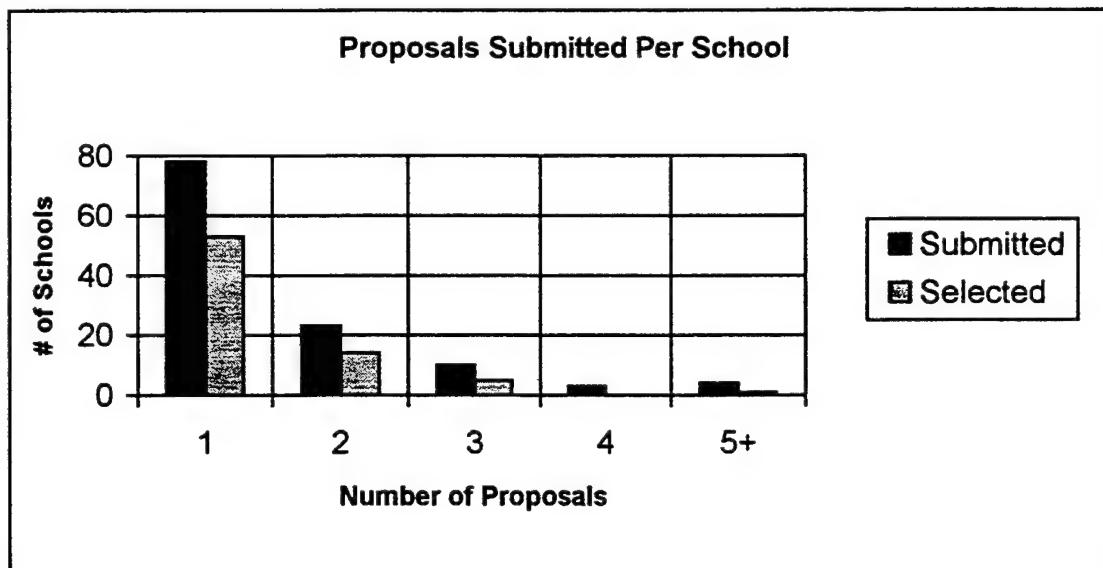
Proposals Submitted and Selected, by Laboratory		
	Applied	Selected
Armstrong Laboratory	41	19
Arnold Engineering Development Center	12	4
Frank J. Seiler Research Laboratory	6	3
Phillips Laboratory	33	19
Rome Laboratory	31	13
Wilford Hall Medical Center	2	1
Wright Laboratory	62	37
TOTAL		

Note: Phillips Laboratory funded 3 SREPs; Wright Laboratory funded 11; and AFOSR funded 11 beyond its contractual 75.

The 306 1994 Summer Research Program participants represented 135 institutions.

Institutions Represented on the 1994 SRP and 1995 SREP		
Number of schools represented in the Summer 92 Program	Number of schools represented in submitted proposals	Number of schools represented in Funded Proposals
135	118	73

Forty schools had more than one participant submitting proposals.



The selection rate for the 78 schools submitting 1 proposal (68%) was better than those submitting 2 proposals (61%), 3 proposals (50%), 4 proposals (0%) or 5+ proposals (25%). The 4 schools that submitted 5+ proposals accounted for 30 (15%) of the 196 proposals submitted.

Of the 196 proposals submitted, 159 offered institution cost sharing. Of the funded proposals which offered cost sharing, the minimum cost share was \$1000.00, the maximum was \$68,000.00 with an average cost share of \$12,016.00.

Proposals and Institution Cost Sharing		
	Proposals Submitted	Proposals Funded
With cost sharing	159	82
Without cost sharing	37	18
Total	196	100

The SREP participants were residents of 41 different states. Number of states represented at each laboratory were:

States Represented, by Proposals Submitted/Selected per Laboratory		
	Proposals Submitted	Proposals Funded
Armstrong Laboratory	21	13
Arnold Engineering Development Center	5	2
Frank J. Seiler Research Laboratory	5	3
Phillips Laboratory	16	14
Rome Laboratory	14	7
Wilford Hall Medical Center	2	1
Wright Laboratory	24	20

Eleven of the 1995 SREP Principal Investigators also participated in the 1994 SREP.

ADMINISTRATIVE EVALUATION: The administrative quality of the SREP associates' final reports was satisfactory. Most complied with the formatting and other instructions provided to them by RDL. Ninety seven final reports and two interim reports have been received and are included in this report. The subcontracts were funded by \$1,991,623.00 of Air Force money. Institution cost sharing totaled \$985,353.00.

TECHNICAL EVALUATION: The form used for the technical evaluation is provided as Appendix 2. ninety-two evaluation reports were received. Participants by laboratory versus evaluations submitted is shown below:

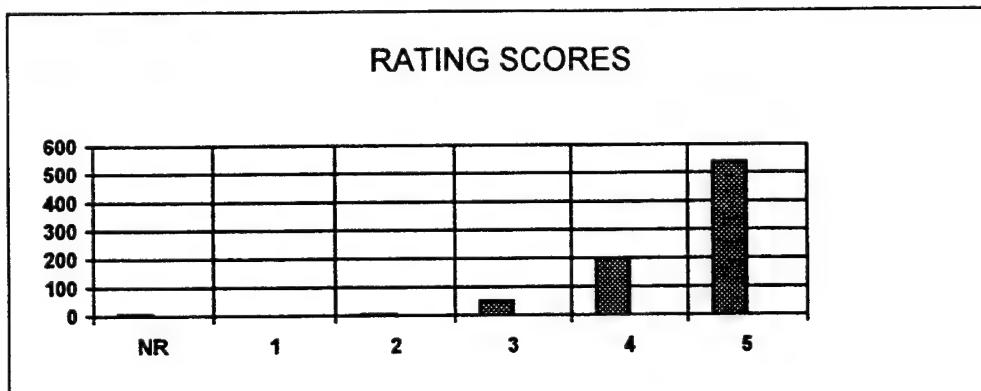
	Participants	Evaluations	Percent
Armstrong Laboratory	23 ¹	20	95.2
Arnold Engineering Development Center	4	4	100
Frank J. Seiler Research Laboratory	3	3	100
Phillips Laboratory	19 ²	18	100
Rome Laboratory	13	13	100
Wilford Hall Medical Center	1	1	100
Wright Laboratory	37	34	91.9
Total			

Notes:

- 1: Research on two of the final reports was incomplete as of press time so there aren't any technical evaluations on them to process, yet. Percent complete is based upon 20/21=95.2%
- 2: One technical evaluation was not completed because one of the final reports was incomplete as of press time. Percent complete is based upon 18/18=100%
- 3: See notes 1 and 2 above. Percent complete is based upon 93/97=95.9%

The number of evaluations submitted for the 1995 SREP (95.9%) shows a marked improvement over the 1994 SREP submittals (65%).

PROGRAM EVALUATION: Each laboratory focal point evaluated ten areas (see Appendix 2) with a rating from one (lowest) to five (highest). The distribution of ratings was as follows:



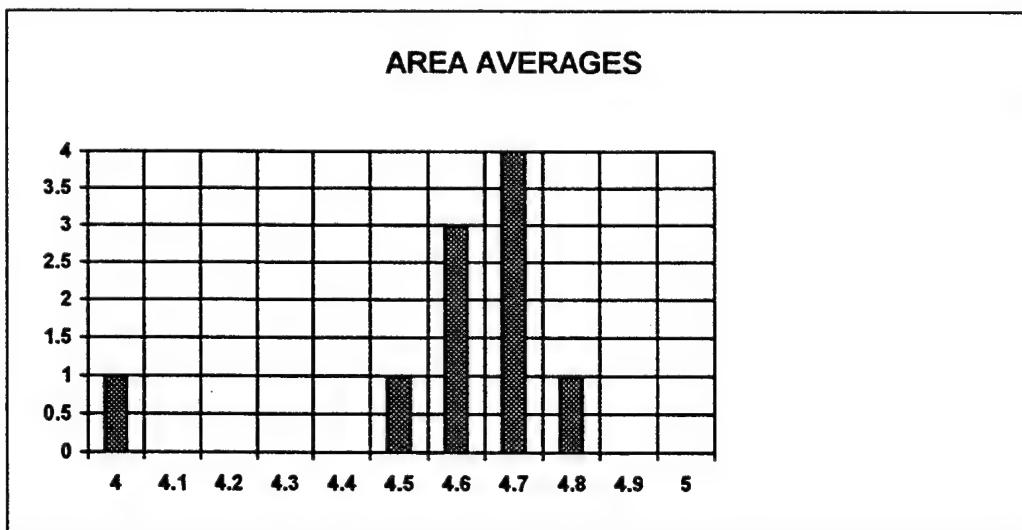
Rating	Not Rated	1	2	3	4	5
# Responses	7	1	7	62 (6%)	226 (25%)	617 (67%)

The 8 low ratings (one 1 and seven 2's) were for question 5 (one 2) "The USAF should continue to pursue the research in this SREP report" and question 10 (one 1 and six 2's) "The

one-year period for complete SREP research is about right", in addition over 30% of the threes (20 of 62) were for question ten. The average rating by question was:

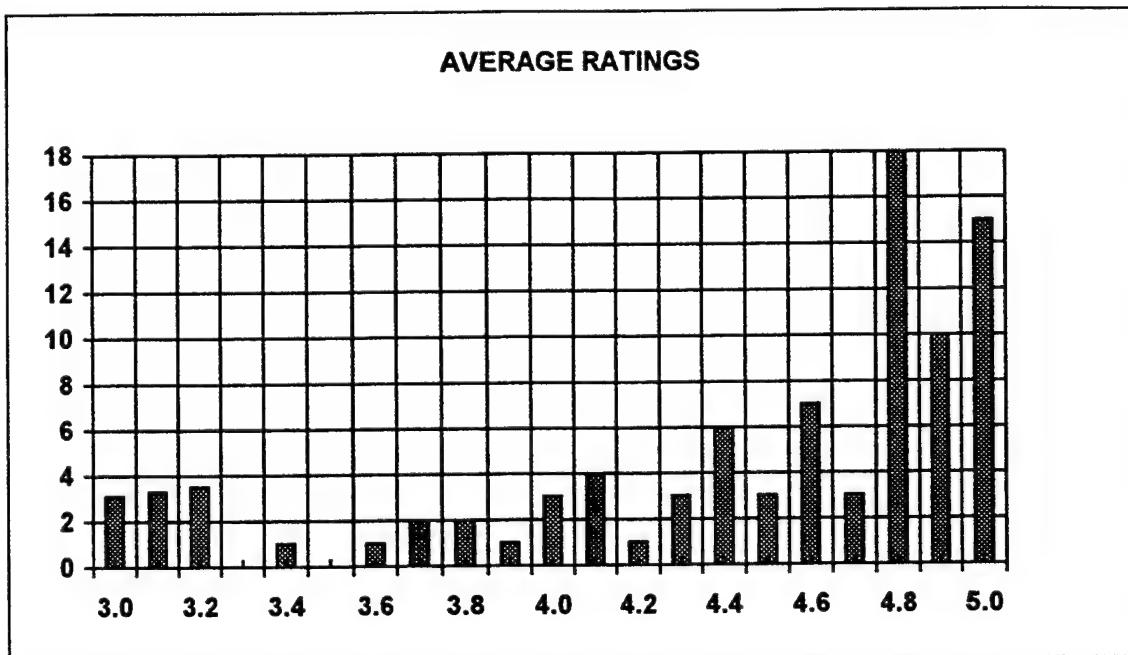
Question	1	2	3	4	5	6	7	8	9	10
Average	4.6	4.6	4.7	4.7	4.6	4.7	4.8	4.5	4.6	4.0

The distribution of the averages was:



Area 10 "the one-year period for complete SREP research is about right" had the lowest average rating (4.1). The overall average across all factors was 4.6 with a small sample standard deviation of 0.2. The average rating for area 10 (4.1) is approximately three sigma lower than the overall average (4.6) indicating that a significant number of the evaluators feel that a period of other than one year should be available for complete SREP research.

The average ratings ranged from 3.4 to 5.0. The overall average for those reports that were evaluated was 4.6. Since the distribution of the ratings is not a normal distribution the average of 4.6 is misleading. In fact over half of the reports received an average rating of 4.8 or higher. The distribution of the average report ratings is as shown:



It is clear from the high ratings that the laboratories place a high value on AFOSR's Summer Research Extension Programs.

3.0 SUBCONTRACTS SUMMARY

Table 1 provides a summary of the SREP subcontracts. The individual reports are published in volumes as shown:

<u>Laboratory</u>	<u>Volume</u>
Armstrong Laboratory	1A, 1B
Arnold Engineering Development Center	5
Frank J. Seiler Research Laboratory	5
Phillips Laboratory	2
Rome Laboratory	3
Wilford Hall Medical Center	5
Wright Laboratory	4A, 4B

1995 SREP SUB-CONTRACT DATA

Report Author Author's University	Author's Degree	Sponsoring Lab	Performance Period	Contract Amount	Univ. Cost Share
Anderson , James Analytical Chemistry University of Georgia, Athens, GA	PhD 95-0807	AL/EQ	01/01/95 12/31/95	\$25000.00	\$1826.00
		Determination of the Redox Capacity of Soil Sediment and Prediction of Pollutant			
Ashrafiun , Hashem Mechanical Engineering Villanova University, Villanova, PA	PhD 95-0800	AL/CF	01/01/95 12/31/95	\$25000.00	\$19528.00
		Finite Element Modeling of the Human Neck and Its Validation for the ATB Model			
Burke , Michael Tulane University Tulane University, New Orleans, LA	PhD 95-0811	AL/HR	01/01/95 09/30/95	\$25000.00	\$1818.00
		An Examination of the Validity of the New Air Force ASVAB Composites			
Edwards , Paul Chemistry Edinboro Univ of Pennsylvania, Edinboro, PA	PhD 95-0808	AL/EQ	01/01/95 12/31/95	\$25000.00	\$5000.00
		Fuel Identification by Neural Networks Analysis of the Response of Vapor Sensiti			
Gerstman , Bernard Physics Florida International Universi, Miami, FL	PhD 95-0815	AL/OE	01/01/95 12/31/95	\$24289.00	\$2874.00
		A Comparison of Multistep vs Singlestep Arrhenius Integral Models for Describing			
Graetz , Kenneth Department of Psychology University of Dayton, Dayton, OH	PhD 95-0812	AL/HR	01/01/95 12/31/95	\$25000.00	\$0.00
		Effects of Mental Workload and Electronic Support on Negotiation Performance			
Gupta , Pushpa Mathematics University of Maine, Orono, ME	PhD 95-0802	AL/AO	01/01/95 12/31/95	\$25000.00	\$2859.00
		Regression to the Mean in Half Life Studies			
Koch , Manfred Geophysics Florida State University, Tallahassee, FL	PhD 95-0809	AL/EQ	12/01/94 04/30/95	\$25000.00	\$0.00
		Application of the MT3D Solute Transport Model to the Made-2 Site: Calibration			
Novotny , Mark Supercomputer Comp Res. I Florida State University, Tallahassee, FL	PhD 95-0810	AL/EQ	01/01/95 12/31/95	\$25000.00	\$0.00
		Computer Calculations of Gas-Phase Reaction Rate Constants			
Nurre , Joseph Mechanical Engineering Ohio University, Athens, OH	PhD 95-0804	AL/CF	01/01/95 12/31/95	\$25000.00	\$20550.00
		Surface Fitting Three Dimensional Human Head Scan Data			
Piepmeyer , Edward Pharmaceutics University of South Carolina, Columbia, SC	PhD 95-0801	AL/AO	01/01/95 12/31/95	\$25000.00	\$11740.00
		The Effects of Hyperbaric Oxygenation on Metabolism of Drugs and Other Xenobioti			
Quinones , Miguel Psychology Rice University, Houston, TX	PhD 95-0813	AL/HR	01/01/95 12/31/95	\$25000.00	\$4000.00
		Maintaining Skills After Training: The Role of Opportunity to Perform Trained T			
Riccio , Gary Psychology Univ of IL Urbana-Champaign, Urbana, IL	PhD 95-0806	AL/CF	01/01/95 05/31/95	\$22931.00	\$0.00
		Nonlinear Transcutaneous Electrical Stimulation of the Vestibular System			
Shebilske , Wayne Dept of Psychology Texas A&M University, College Station, TX	PhD 95-0814	AL/HR	01/01/95 12/31/95	\$25000.00	\$5614.00
		Cognitive Factors in Distr Training Effects During Acquisition of Complex Skills			

1995 SREP SUB-CONTRACT DATA

Report Author Author's University	Author's Degree	Sponsoring Lab	Performance Period	Contract Amount	Univ. Cost Share
Weisenberger , Janet Dept of Speech & Hearing Ohio State University, Columbus, OH	PhD 95-0805	AL/CF	01/01/95 12/31/95	\$25000.00	\$12234.00
Hughes , Rod Psychology Oregon Health Sciences University, Portland, OR	MA 95-0803		Tactile Feedback for Simulation of Object Shape and Textural Information in Hapt		
Bapty , Theodore Electrical Engineering Vanderbilt University, Nashville, TN	MS 95-0848	AEDC/E	01/01/95 12/31/95	\$24979.00	\$0.00
Dorgan , John Chemical Engineering Colorado School of Mines, Golden, CO	PhD 95-0834	FJSRL/F	01/01/95 12/31/95	\$25000.00	\$0.00
Jungbauer , Mary Ann Chemistry Barry University, Miami, FL	PhD 95-0836	FJSRL/F	01/01/95 12/31/95	\$25000.00	\$24714.00
Statman , David Physics Allegheny College, Meadville, PA	PhD 95-0835	FJSRL/F	01/01/95 12/31/95	\$25000.00	\$6500.00
, Krishnaswamy Aeronautics University of Houston, Houston, TX	PhD 95-0818	PL/RK	01/01/95 12/31/95	\$24993.00	\$8969.00
Ashgriz , Nasser Mechanical Engineering SUNY-Buffalo, Buffalo, NY	PhD 95-0816	PL/RK	01/01/95 12/31/95	\$25000.00	\$22329.00
Bellem , Raymond Computer Science Embry-Riddle Aeronautical Univ, Prescott, AZ	PhD 95-0817	PL/VT	12/01/94 11/30/95	\$20000.00	\$8293.00
Brzosko , Jan Nuclear Physics Stevens Institute of Tech, Hoboken, NJ	PhD 95-0828	PL/WS	11/01/94 02/01/95	\$24943.00	\$0.00
Damodaran , Meledath Math & Computer Science University of Houston-Victoria, Victoria, TX	PhD 95-0831	PL/LI	01/01/95 12/31/94	\$24989.00	\$9850.00
DeLyser , Ronald Electrical Engineering University of Denver, Denver, CO	PhD 95-0877	PL/WS	01/01/95 12/31/95	\$25000.00	\$46066.00
Diels , Jean-Claude Physics University of New Mexico, Albuquerque, NM	PhD 95-0819	PL/LI	01/01/95 12/31/95	\$25000.00	\$0.00
Henson , James Electrical Engineering University of Nevada, Reno, NV	PhD 95-0820	PL/WS	01/01/95 12/31/95	\$25000.00	\$0.00
Kaiser , Gerald Physics University of Mass/Lowell, Lowell, MA	PhD 95-0821	PL/GP	01/01/95 12/31/95	\$25000.00	\$5041.00
			Multiresolution Analysis with Physical Wavelets		

1995 SREP SUB-CONTRACT DATA

Report Author Author's University	Author's Degree	Sponsoring Lab	Performance Period	Contract Amount	Univ. Cost Share
Kowalak , Albert Chemistry University of Massachusetts/Lo, Lowell, MA	PhD 95-0822	PL/GP	01/01/95 12/31/95	\$24996.00	\$4038.00
Malloy , Kevin Electrical Engineering University of New Mexico, Albuquerque, NM	PhD 95-0829	PL/VT	01/01/95 12/31/95	\$24999.00	\$0.00
Prasad , Sudhakar Physics University of New Mexico, Albuquerque, NM	PhD 95-0823	PL/LI	01/01/95 12/31/95	\$25000.00	\$11047.00
Purtill , Mark Mathematics Texas A&M Univ-Kingsville, Kingsville, TX	PhD 95-0824	PL/WS	01/01/95 12/31/95	\$25000.00	\$100.00
Rudolph , Wolfgang Physics University of New Mexico, Albuquerque, NM	PhD 95-0833	PL/LI	01/01/95 12/31/95	\$24982.00	\$6000.00
Stone , Alexander Mathematics & Statistics University of New Mexico, Alburquerque, NM	PhD 95-0827	PL/WS	01/01/95 12/31/95	\$24969.00	\$0.00
Swenson , Charles Dept of Electrical Engr Utah State University, Logan, UT	PhD 95-0826	PL/VT	01/01/95 12/31/95	\$25000.00	\$25000.00
Lipp , John Electrical Engineering Michigan Technological Univ, Houghton, MI	MS 95-0832	PL/LI	01/01/95 12/31/95	\$24340.00	\$15200.00
Petroski , Janet Chemistry Cal State Univ/Northridge, Northridge, CA	BA 95-0830	PL/RK	10/01/94 12/31/94	\$4279.00	\$0.00
Salasovich , Richard Mechanical Engineering University of Cincinnati, Cincinnati, OH	MS 95-0825	PL/VT	01/01/95 12/31/95	\$25000.00	\$4094.00
Aalo , Valentine Dept of Electrical Engr Florida Atlantic University, Boca Raton, FL	PhD 95-0837	RL/C3	01/01/95 12/31/95	\$25000.00	\$13120.00
Amin , Moeness Electrical Engineering Villanova University, Villanova, PA	PhD 95-0838	RL/C3	01/01/95 12/31/95	\$25000.00	\$34000.00
Benjamin , David Computer Science Oklahoma State University, Stillwater, OK	PhD 95-0839	RL/C3	01/01/95 12/31/95	\$24970.00	\$0.00
Choudhury , Ajit Engineering Howard University, Washington, DC	PhD 95-0840	RL/OC	11/30/94 10/31/95	\$25000.00	\$0.00
Harackiewicz , Frances Electrical Engineering So. Illinois Univ-Carbondale, Carbondale, IL	PhD 95-0841	RL/ER	01/01/95 12/31/95	\$23750.00	\$29372.00

1995 SREP SUB-CONTRACT DATA

Report Author Author's University	Author's Degree	Sponsoring Lab	Performance Period	Contract Amount	Univ. Cost Share
Losiewicz , Beth Psycholinguistics Colorado State University, Fort Collins, CO	PhD 95-0842	RL/IR	01/01/95 12/31/95	\$25000.00	\$4850.00
Musavi , Mohamad University of Maine, Orono, ME	PhD 95-0843	RL/IR	01/01/95 12/31/95	\$25000.00	\$12473.00
Norgard , John Elec & Comp Engineering Univ of Colorado-Colorado Sprg, Colorado	PhD 95-0844	RL/ER	01/01/95 12/31/95	\$25000.00	\$2500.00
Richardson , Dean Photonics SUNY Institute of Technology, Utica, NY	PhD 95-0845	RL/OC	01/01/95 12/31/95	\$25000.00	\$15000.00
Ryder, Jr. , Daniel Chemical Engineering Tufts University, Medford, MA	PhD 95-0846	RL/ER	01/01/95 12/31/95	\$25000.00	\$0.00
Zhang , Xi-Cheng Physics Rensselaer Polytechnic Institu, Troy, NY	PhD 95-0847	RL/ER	01/01/95 12/31/95	\$25000.00	\$0.00
Drost-Hansen , Walter Chemistry University of Miami, Coral Gables, FL	PhD 95-0875	WHMC/	01/01/95 12/31/95	\$25000.00	\$8525.00
Baginski , Michael Electrical Engineering Auburn University, Auburn, AL	PhD 95-0869	WL/MN	01/01/95 12/31/95	\$24995.00	\$10098.00
Berdichevsky , Victor Aerospace Engineering Wayne State University, Detroit, MI	PhD 95-0849	WL/FI	01/01/95 12/31/95	\$25000.00	\$0.00
Buckner , Steven Chemistry Colullmbus College, Columbus, GA	PhD 95-0850	WL/PO	01/01/95 12/31/95	\$24900.00	\$8500.00
Carroll , James Electrical Engineering Clarkson University, Potsdam, NY	PhD 95-0881	WL/PO	01/01/95 12/31/95	\$24944.00	\$38964.00
Choate , David Mathematics Transylvania University, Lexington, KY	PhD 95-0851	WL/AA	01/01/95 12/31/95	\$24993.00	\$8637.00
Clarson , Stephen Materials Sci & Eng University of Cincinnati, Cincinnati, OH	PhD 95-0852	WL/ML	12/01/94 11/30/95	\$25000.00	\$15000.00
Cone , Milton Comp Science & Elec Eng Embry-Riddel Aeronautical Univ, Prescott, AZ	PhD 95-0853	WL/AA	01/01/95 12/31/95	\$25000.00	\$11247.00
Courter , Robert Mechanical Engineering Louisiana State University, Baton Rouge, LA	PhD 95-0854	WL/MN	01/01/95 12/31/95	\$25000.00	\$3729.00

1995 SREP SUB-CONTRACT DATA

Report Author Author's University	Author's Degree	Sponsoring Lab	Performance Period	Contract Amount	Univ. Cost Share
Dominic , Vincent Electro Optics Program University of Dayton, Dayton, OH	PhD 95-0868	WL/ML	01/01/95 12/31/95	\$25000.00	\$12029.00
			Characterization of Electro-Optic Polymers		
Fadel , Georges Dept of Mechanical Engr Clemson University, Clemson, SC	PhD 95-0855	WL/MT	01/01/95 12/31/95	\$25000.00	\$8645.00
			A Methodology for Affordability in the Design Process		
Gould , Richard Mechanical Engineering North Carolina State Univ, Raleigh, NC	PhD 95-0856	WL/PO	01/01/95 12/31/95	\$24998.00	\$9783.00
			Data Reduction and Analysis for laser Doppler Velocimetry		
Hardie , Russell Electrical Engineering Unvisty of Dayton, Dayton, OH	PhD 95-0882	WL/AA	01/01/95 12/31/95	\$24999.00	\$7415.00
			Hyperspectral Target Identification Using Bomen Spectrometer Data		
Hodel , Alan Electrical Engineering Auburn University, Auburn, AL	PhD 95-0870	WL/MN	01/01/95 12/31/95	\$24990.00	\$9291.00
			Robust Falut Tolerant Control: Fault Detection and Classification		
Janus , Jonathan Aerospace Engineering Mississippi State University, Mississippi State,	PhD 95-0871	WL/MN	01/01/95 12/31/95	\$25000.00	\$7143.00
			Multidimensional Algorithm Development & Analysis		
Jasiuk , Iwona Dept of Materials Science Michigan State University, East Lansing, MI	PhD 95-0857	WL/ML	01/01/95 12/31/95	\$25000.00	\$0.00
			Characterization of Interfaces in Metal-Matrix Composites		
Jouny , Ismail Electrical Engineering Lafayette College, Easton, PA	PhD 95-0880	WL/AA	01/01/95 12/31/95	\$24300.00	\$5200.00
			TSI Mitigation: A Mountaintop Database Study		
Li , Jian Electrical Engineering University of Florida, Gainesville, FL	PhD 95-0859	WL/AA	10/10/95 12/31/95	\$25000.00	\$4000.00
			Comparative Study and Performance Analysis of High Resolution SAR Imaging Techni		
Lin , Chun-Shin Electrical Engineering University of Missouri-Columbi, Columbia, MO	PhD 95-0883	WL/FI	01/01/95 12/31/95	\$25000.00	\$2057.00
			Prediction of Missile Trajectory		
Lin , Paul Mechanical Engineering Cleveland State University, Cleveland, OH	PhD 95-0860	WL/FI	01/01/95 12/31/95	\$25000.00	\$6886.00
			Three Dimensional Deformation Comparison Between Bias and Radial Aircraft Tires		
Liou , Juin Electrical Engineering University of Central Florida, Orlando, FL	PhD 95-0876	WL/EL	01/01/95 12/31/95	\$25000.00	\$11040.00
			Investigation of AlGaAs/GaAs Heterojunction Bipolar Transister Reliability Based		
Nandhakumar , Nagaraj Electrical Engineering University of Virginia, Charlottesville, VA	PhD 95-0861	WL/AA	01/01/95 12/31/95	\$24979.00	\$4500.00
			Thermophysical Invariants fro, LWIR Imagery for ATR		
Pasala , Krishna Dept of Electrical Engr University of Dayton, Dayton, OH	PhD 95-0879	WL/AA	01/01/95 12/31/95	\$25000.00	\$1078.00
			Effect of Electromagmetic Enviorment on Array Signal Processing		
Perkowski , Marek Dept of Electrical Engr Portland State University, Portland, OR	PhD 95-0878	WL/AA	01/01/95 09/15/95	\$24947.00	\$18319.00
			Functional Decomposition of Binary, Multiple-Valued, & Fuzzy Logic		

1995 SREP SUB-CONTRACT DATA

Report Author Author's University	Author's Degree	Sponsoring Lab	Performance Period	Contract Amount	Univ. Cost Share
Reeves , Stanley Dept of Electrical Engr Auburn University, Auburn, AL	PhD 95-0862	WL/MN	01/01/95 12/31/95	\$25000.00	\$0.00
			Superresolution of Passive Imaging	Millimeter-Wave	
Rule , William Engineering Mechanics University of Alabama, Tuscaloosa, AL	PhD 95-0872	WL/MN	01/01/95 12/31/95	\$24968.00	\$14576.00
			Development of a Penetrator Optimizer		
Schauer , John Mech & Aerosp Eng University of Dayton, Dayton, OH	PhD 95-0873	WL/PO	11/01/94 11/30/95	\$25000.00	\$7428.00
			Heat Transfer for Turbine Blade Film Cooling		
			with Free Stream Turbulence - Measu		
Schwartz , Carla Electrical Engineering University of Florida, Gainesville, FL	PhD 95-0863	WL/FI	01/01/95 12/31/95	\$25000.00	\$0.00
			Neural Network Identification and Control in		
			Metal Forging		
Simon , Terrence Dept of Mechanical Engineering University of Minnesota, Minneapolis, MN	PhD 95-0864	WL/PO	01/01/95 12/31/95	\$24966.00	\$3996.00
			Documentation of Separating and Separated		
			Boundary Layer Flow, for Application		
Skowronski , Marek Solid State Physics Carnegie Melon University, Pittsburgh, PA	PhD 95-0865	WL/EL	01/01/95 12/31/95	\$25000.00	\$6829.00
			Transmission Electron Microscopy of		
			Semiconductor Heterojunctions		
Thirunarayan , Krishnaprasad Computer Science Wright State University, Dayton, OH	PhD 95-0866	WL/EL	01/01/95 12/31/95	\$25000.00	\$2816.00
			VHDL-93 Parser in SWI-PROLOG: A Basis for Design		
			Query System		
Trelease , Robert Dept of Anatomy & Cell Bi University of California, Los Angeles, CA	PhD 95-0867	WL/ML	12/01/94 12/01/95	\$25000.00	\$0.00
			Development of Qualitative Process Control		
			Discovery Systems for Polymar Composi		
Tsai , Chi-Tay Engineering Mechanics Florida Atlantic University, Boca Raton, FL	PhD 95-0874	WL/MN	01/01/95 12/31/95	\$24980.00	\$0.00
			Improved Algorithm Development of Massively		
			Parallel Epic Hydrocode in Cray T3D		
Lewis , John Materials Science Engrng University of Kentucky, Lexington, KY	MS 95-0858	WL/MN	01/01/95 12/31/95	\$25000.00	\$13833.00
			The Characterization of the Mechanical		
			Properties of Materials in a Biaxial Stre		

APPENDIX 1:
SAMPLE SREP SUBCONTRACT

**AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
1995 SUMMER RESEARCH EXTENSION PROGRAM
SUBCONTRACT 95-0837**

BETWEEN

Research & Development Laboratories
5800 Uplander Way
Culver City, CA 90230-6608

AND

Florida Atlantic University
Department of Electrical Engineering
Boca Raton, FL 33431

REFERENCE: Summer Research Extension Program Proposal 95-0837
Start Date: 01-01-95 End Date: 12-31-95
Proposal Amount: \$25,000.00

(1) **PRINCIPAL INVESTIGATOR:** Dr. Valentine A. Aalo
Department of Electrical Engineering
Florida Atlantic University
Boca Raton, FL 33431

(2) **UNITED STATES AFOSR CONTRACT NUMBER:** F49620-93-C-0063

(3) **CATALOG OF FEDERAL DOMESTIC ASSISTANCE NUMBER (CFDA):12.800**
PROJECT TITLE: AIR FORCE DEFENSE RESEARCH SOURCES PROGRAM

(4) **ATTACHMENT 1 REPORT OF INVENTIONS AND SUBCONTRAACT**
 2 **CONTRACT CLAUSES**
 3 **FINAL REPORT INSTRUCTIONS**

*****SIGN SREP SUBCONTRACT AND RETURN TO RDL*****

1. BACKGROUND: Research & Development Laboratories (RDL) is under contract (F49620-93-C-0063) to the United States Air Force to administer the Summer Research Program (SRP), sponsored by the Air Force Office of Scientific Research (AFOSR), Bolling Air Force Base, D.C. Under the SRP, a selected number of college faculty members and graduate students spend part of the summer conducting research in Air Force laboratories. After completion of the summer tour participants may submit, through their home institutions, proposals for follow-on research. The follow-on research is known as the Summer Research Extension Program (SREP). Approximately 61 SREP proposals annually will be selected by the Air Force for funding of up to \$25,000; shared funding by the academic institution is encouraged. SREP efforts selected for funding are administered by RDL through subcontracts with the institutions. This subcontract represents an agreement between RDL and the institution herein designated in Section 5 below.

2. RDL PAYMENTS: RDL will provide the following payments to SREP institutions:

- 80 percent of the negotiated SREP dollar amount at the start of the SREP research period.
- The remainder of the funds within 30 days after receipt at RDL of the acceptable written final report for the SREP research.

3. INSTITUTION'S RESPONSIBILITIES: As a subcontractor to RDL, the institution designated on the title page will:

- a. Assure that the research performed and the resources utilized adhere to those defined in the SREP proposal.
- b. Provide the level and amounts of institutional support specified in the SREP proposal..
- c. Notify RDL as soon as possible, but not later than 30 days, of any changes in 3a or 3b above, or any change to the assignment or amount of participation of the Principal Investigator designated on the title page.
- d. Assure that the research is completed and the final report is delivered to RDL not later than twelve months from the effective date of this subcontract, but no later than December 31, 1998. The effective date of the subcontract is one week after the date that the institution's contracting representative signs this subcontract, but no later than January 15, 1998.
- e. Assure that the final report is submitted in accordance with Attachment 3.
- f. Agree that any release of information relating to this subcontract (news releases, articles, manuscripts, brochures, advertisements, still and motion pictures, speeches, trade associations meetings, symposia, etc.) will include a statement that the project or effort depicted was or is sponsored by: Air Force Office of Scientific Research, Bolling AFB, D.C.
- g. Notify RDL of inventions or patents claimed as the result of this research as specified in Attachment 1.

- h. RDL is required by the prime contract to flow down patent rights and technical data requirements to this subcontract. Attachment 2 to this subcontract

contains a list of contract clauses incorporated by reference in the prime contract.

4. All notices to RDL shall be addressed to:

RDL AFOSR Program Office
5800 Uplander Way
Culver City, CA 90230-6609

5. By their signatures below, the parties agree to provisions of this subcontract.



Abe Sopher
RDL Contracts Manager

Signature of Institution Contracting Official

Typed/Printed Name

Date

Title

Institution

Date/Phone

ATTACHMENT 2
CONTRACT CLAUSES

This contract incorporates by reference the following clauses of the Federal Acquisition Regulations (FAR), with the same force and effect as if they were given in full text. Upon request, the Contracting Officer or RDL will make their full text available (FAR 52.252-2).

<u>FAR CLAUSES</u>	<u>TITLE AND DATE</u>
52.202-1	DEFINITIONS
52.203-3	GRATUITIES
52.203-5	COVENANT AGAINST CONTINGENT FEES
52.203-6	RESTRICTIONS ON SUBCONTRACTOR SALES TO THE GOVERNMENT
52.203-7	ANTI-KICKBACK PROCEDURES
52.203-8	CANCELLATION, RECISSION, AND RECOVERY OF FUNDS FOR ILLEGAL OR IMPROPER ACTIVITY
52.203-10	PRICE OR FEE ADJUSTMENT FOR ILLEGAL OR IMPROPER ACTIVITY
52.203-12	LIMITATION ON PAYMENTS TO INFLUENCE CERTAIN FEDERAL TRANSACTIONS
52.204-2	SECURITY REQUIREMENTS
52.209-6	PROTECTING THE GOVERNMENT'S INTEREST WHEN SUBCONTRACTING WITH CONTRACTORS DEBARRED, SUSPENDED, OR PROPOSED FOR DEBARMENT
52.212-8	DEFENSE PRIORITY AND ALLOCATION REQUIREMENTS
52.215-2	AUDIT AND RECORDS - NEGOTIATION
52.215-10	PRICE REDUCTION FOR DEFECTIVE COST OR PRICING DATA

52.215-12	SUBCONTRACTOR COST OR PRICING DATA
52.215-14	INTEGRITY OF UNIT PRICES
52.215-8	ORDER OF PRECEDENCE
52.215.18	REVERSION OR ADJUSTMENT OF PLANS FOR POSTRETIREMENT BENEFITS OTHER THAN PENSIONS
52.222-3	CONVICT LABOR
52.222-26	EQUAL OPPORTUNITY
52.222-35	AFFIRMATIVE ACTION FOR SPECIAL DISABLED AND VIETNAM ERA VETERANS
52.222-36	AFFIRMATIVE ACTION FOR HANDICAPPED WORKERS
52.222-37	EMPLOYMENT REPORTS ON SPECIAL DISABLED VETERAN AND VETERANS OF THE VIETNAM ERA
52.223-2	CLEAN AIR AND WATER
52.223-6	DRUG-FREE WORKPLACE
52.224-1	PRIVACY ACT NOTIFICATION
52.224-2	PRIVACY ACT
52.225-13	RESTRICTIONS ON CONTRACTING WITH SANCTIONED PERSONS
52.227-1	ALT. I - AUTHORIZATION AND CONSENT
52.227-2	NOTICE AND ASSISTANCE REGARDING PATIENT AND COPYRIGHT INFRINGEMENT

52.227-10	FILING OF PATENT APPLICATIONS - CLASSIFIED SUBJECT MATTER
52.227-11	PATENT RIGHTS - RETENTION BY THE CONTRACTOR (SHORT FORM)
52.228-7	INSURANCE - LIABILITY TO THIRD PERSONS
52.230-5	COST ACCOUNTING STANDARDS - EDUCATIONAL INSTRUCTIONS
52.232-23	ALT. I - ASSIGNMENT OF CLAIMS
52.233-1	DISPUTES
52.233-3	ALT. I - PROTEST AFTER AWARD
52.237-3	CONTINUITY OF SERVICES
52.246-25	LIMITATION OF LIABILITY - SERVICES
52.247-63	PREFERENCE FOR U.S. - FLAG AIR CARRIERS
52.249-5	TERMINATION FOR CONVENIENCE OF THE GOVERNMENT (EDUCATIONAL AND OTHER NONPROFIT INSTITUTIONS)
52.249-14	EXCUSABLE DELAYS
52.251-1	GOVERNMENT SUPPLY SOURCES

<u>DOD FAR CLAUSES</u>	<u>DESCRIPTION</u>
252.203-7001	SPECIAL PROHIBITION ON EMPLOYMENT
252.215-7000	PRICING ADJUSTMENTS
252.233-7004	DRUG FREE WORKPLACE (APPLIES TO SUBCONTRACTS WHERE THERE IS ACCESS TO CLASSIFIED INFORMATION)
252.225-7001	BUY AMERICAN ACT AND BALANCE OF PAYMENTS PROGRAM
252.225-7002	QUALIFYING COUNTRY SOURCES AS SUBCONTRACTS
252.227-7013	RIGHTS IN TECHNICAL DATA - NONCOMMERCIAL ITEMS
252.227-7030	TECHNICAL DATA - WITHHOLDING PAYMENT
252.227-7037	VALIDATION OF RESTRICTIVE MARKINGS ON TECHNICAL DATA
252.231-7000	SUPPLEMENTAL COST PRINCIPLES
252.232-7006	REDUCTIONS OR SUSPENSION OF CONTRACT PAYMENTS UPON FINDING OF FRAUD

APPENDIX 2:
SAMPLE TECHNICAL EVALUATION FORM

SUMMER RESEARCH EXTENSION PROGRAMTECHNICAL EVALUATION

SREP NO: 95-0811

SREP PRINCIPAL INVESTIGATOR: Dr. Michael Burke

Circle the rating level number, 1 (low) through 5 (high), you feel best evaluate each statement and return thecompleted form by mail to:

RDL
Attn: 1995 SREP Tech Evals
5800 Uplander Way
Culver City, CA 90230-6608
(310) 216-5940 or (800) 677-1363

1. This SREP report has a high level of technical merit. 1 2 3 4 5
2. The SREP program is important to accomplishing the lab's mission. 1 2 3 4 5
3. This SREP report accomplished what the associate's proposal promised. 1 2 3 4 5
4. This SREP report addresses area(s) important to the USAF. 1 2 3 4 5
5. The USAF should continue to pursue the research in this SREP report. 1 2 3 4 5
6. The USAF should maintain research relationships with this SREP associate. 1 2 3 4 5
7. The money spent on this SREP effort was well worth it. 1 2 3 4 5
8. This SREP report is well organized and well written. 1 2 3 4 5
9. I'll be eager to be a focal point for summer and SREP associates in the future. 1 2 3 4 5
10. The one-year period for complete SREP research is about right. 1 2 3 4 5

11. If you could change any one thing about the SREP program, what would you change.

12. What would you definitely NOT change about the SREP program?

USE THE BACK FOR ANY ADDITIONAL COMMENTS.

Laboratory: Armstrong Laboratory

Lab Focal Point: Linda Sawin Office Symbol: AL/HRMI

Phone: (210) 536-3876

PREDICTION OF MISSILE TRAJECTORY

Chun-Shin Lin
Associate Professor
Department of Electrical Engineering

University of Missouri-Columbia
Columbia, MO 65211
Tel.: (314)882-3520
e-mail address: eclin@mizzou1.missouri.edu

Final Report for :
Summer Faculty Research Program
Wright Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Wright Laboratory

December 1995

PREDICTION OF MISSILE TRAJECTORY

Chun-Shin Lin

Associate Professor

Department of Electrical and Computer Engineering
University of Missouri-Columbia

Abstract

This study concerns the survivability of an airship that is under attack by a missile. The pilot may employ an appropriate maneuver or countermeasure at the right moment to escape from the attack. Real-time prediction of the missile trajectory will help prepare the pilot to take action at the right moment to survive. The considered missiles have a boosting period of 5-10 seconds followed by a non-powered period. One of our previous study investigated the prediction of missile trajectory only for the non-powered period. A small amount of trajectory data after the motor burns out was used for predicting the future missile trajectory. While the missile-target impact can happen during or shortly after the boosting period, it is necessary to start the prediction at the time when the missile is detected. This extension research investigates the technique for such usage. The technique takes new missile trajectory data once a second for predicting the future missile speed and estimating the navigation constant. With the speed curve and navigation constant value obtained, prediction of the missile trajectory is made. The short computational time of the procedure makes practical implementation more feasible and attractive. Twenty seven sets of trajectory data, nine sets from each of three different types of missiles, are available for this study. Some of them have the impacts happen during the boosting period and some after. In six of them, the target uses high maneuvering as countermeasure. A sliding sample window is used for continually updating the prediction. Prediction errors on the time of impact and the location, for each sample window, are provided. Selected trajectory and error plots are included in this report.

SUMMARY OF NOTATIONS

t :	current time
$\mathbf{V}_M(t)$:	missile velocity vector at t
$V_M(t)$:	missile speed at t
$\mathbf{V}_T(t)$:	target velocity vector at t
$V_T(t)$:	target speed at t
$\mathbf{P}_M(t)$:	missile position vector at t
$\mathbf{P}_T(t)$:	target position vector at t
$\mathbf{R}_{TM}(t)$:	line-of-sight vector at t ; it equals $\mathbf{P}_T(t) - \mathbf{P}_M(t)$
$R_{TM}(t)$:	length of the vector $\mathbf{R}_{TM}(t)$
$\mathbf{V}_{TM}(t)$:	relative velocity between the target and the missile at t ; it equals $\mathbf{V}_T(t) - \mathbf{V}_M(t)$
$V_c(t)$:	closing rate ($= -\dot{R}_{TM}(t)$)
$A_c(t)$:	command acceleration for the missile at t
$\omega_s(t)$:	derivative of line-of-sight angle at t (called the line-of-sight rate)
M :	mass of the missile; the variable cancels out in our final equations
f_{ric} :	friction parameter of the missile; estimated from the missile trajectory data
α :	actual navigation constant
$\hat{\alpha}_i$:	navigation constant estimated at time t_i
$\hat{\alpha}$:	a weighted average of $\hat{\alpha}_i$'s as an estimate of α .
g :	gravity (32.15 ft/sec ²)
$h(t)$:	altitude of the missile at t ; the third component of $\mathbf{P}_M(t)$.
Δt :	sampling period for missile and target trajectory data (50 ms in this study)
\mathbf{k}_N :	the unit vector that is perpendicular to $\mathbf{R}_{TM}(t)$ and lies on the plane consisting of $\mathbf{R}_{TM}(t)$ and $\mathbf{R}_{TM}(t+\Delta t)$

(A symbol with $\hat{}$ on it denotes a predicted or estimated quantity or vector)

PREDICTION OF MISSILE TRAJECTORY

Chun-Shin Lin

1. Introduction

This study concerns the survivability of an airship that is under attack by a missile. The pilot may employ an appropriate maneuver or countermeasure at the right moment to escape from the attack. Real-time prediction of the missile trajectory will help provide needed information earlier to give enough time for the pilot or the electronic decision aid to choose the best action [1].

The considered missiles have a boosting period of 5-10 seconds followed by a non-powered period. The speed of the missile may reach a value greater than 4,000 ft/sec at the end of the boosting period. With such a high speed, the missile can continue to chase its target until the speed drops to too low due to friction. In our previous study [2], we considered the prediction of missile trajectory only for the non-powered (after boosting) period. A small amount of trajectory data after the motor burns out was used for predicting the future missile trajectory. While the missile-target impact can happen during or shortly after the boosting period, it is desired that the prediction starts at the time when the missile is detected. This extension research investigates the technique for such usage.

In this report, the prediction technique and results are presented. In Section 2, methods for prediction of the missile speed and estimation of the navigation constant are presented. Based on these, the prediction procedure is developed and introduced in Section 3. Section 4 gives the experimental results. A relevant problem is to guess which airship is the most likely target of a missile if there are many airships around. This can be done by carefully evaluating the estimated navigation gain for each airship. A technique for identifying the target from trajectory data is reported in Section 5. Section 6 gives the conclusion.

2. Speed Curve and Control Technique of a Missile

Prediction of a missile trajectory requires the knowledge on

- how the missile speed will change from time to time, and
- how the heading direction of the missile will change in order to chase a target

The former is mainly determined by the acceleration during the boosting period, and the friction after. Figure 1 shows a typical missile speed curve. The latter is determined by the control scheme. It is known that today's missiles use the proportional navigation guidance [3]. The most important parameter in this technique is the navigation constant. If the future speed can be predicted and the navigation constant can be estimated, the missile trajectory is predictable.

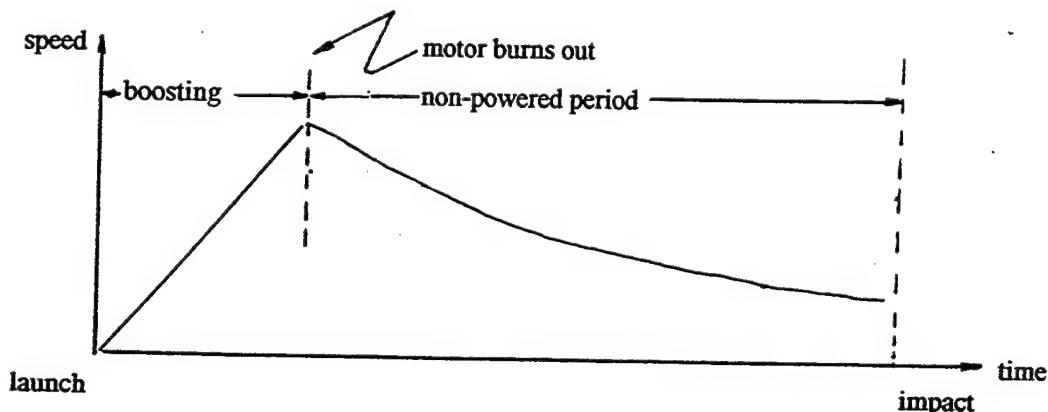


Figure 1. A typical speed curve for a missile

2.1 Prediction of the Future Speed

There are two segments in the speed curve - the acceleration segment and the deceleration segment. Before the motor burns out, a prediction must be for both segments. However, a prediction made after the boosting period will be for the deceleration segment only. Prediction for both segments is a more difficult problem because more unknown factors (burnout time, friction coefficient, etc.) may involve. Methods used for prediction in both cases are discussed below.

Prediction of the Deceleration Segment

Prediction of the deceleration segment is required if the prediction is made after the boosting period. The major factor affecting the speed is friction. In our technique, the friction coefficient is assumed to be a constant and the following relationship is used:

$$V_M(t+\Delta t) = V_M(t) + fric * V_M(t) * \Delta t. \quad (1)$$

where "fric" is referred to as the friction parameter and has a negative value.

The possible change of the missile altitude will affect the velocity. The change of the potential energy will be converted into the kinetic energy. Thus the velocity change from this factor can be solved from the following relationship:

$$\frac{1}{2} [(M V_M^2(t+\Delta t) - M V_M^2(t))] = - Mg[h(t+\Delta t) - h(t)] \quad (2)$$

where M is the mass of the missile, g equals to 32.15 ft/sec^2 , and $h(t)$ is the altitude of the missile at time t . With $V_M(t+\Delta t) \approx V_M(t) + \Delta V_M(t)$, the following can be derived from (2):

$$\Delta V_M(t) \approx g[h(t+\Delta t) - h(t)]/V_M(t) \quad (3)$$

Note that the mass in (2) cancels out and does not appear in (3). With this factor added, (1) can be modified as

$$V_M(t+\Delta t) = V_M(t) + fric * V_M(t) * \Delta t + g[h(t+\Delta t) - h(t)]/V_M(t). \quad (4)$$

With $V_M(t+\Delta t)$, $V_M(t)$, $h(t+\Delta t)$, and $h(t)$ given, $fric$ can be computed from (4). Thus, theoretically, only the trajectory data at two time instants, t and $t+\Delta t$, are needed to evaluate this parameter. However, with possible measurement errors, the average value computed over a longer time period is preferred. If the altitude and velocity of the missile at n consecutive sampled time instants are available, one can compute the mean value of $fric$ solved from (4) as

$$fric = (1/n) \sum_{k=1}^n \{ V_M(t+k\Delta t) - V_M(t+(k-1)\Delta t) - g[h(t+k\Delta t) - h(t+(k-1)\Delta t)] / [V_M(t+(k-1)\Delta t) * \Delta t] \} / [V_M(t+(k-1)\Delta t) * \Delta t] \quad (5)$$

With $fric$ estimated from (5), future speed can be calculated from time to time using (4) or the simplified equation in (1).

Prediction of Both Segments

Prediction for both segments is necessary if it is made before the motor burns out. Using past missile velocity data, a linear regression can be done to obtain the predicted straight-line speed segment. Although the actual acceleration is not a constant, experiments show that the straight-line approximation is good enough for our purpose. Two questions need to be answered before one can make the complete speed prediction; one is when the motor will burn out and the other is what will be the friction after burnout. Without knowing the missile type, one possible solution is to make the best guess on the maximum speed and the friction parameter. Average or medium values for various types of missiles can be used. With the maximum speed given, one can determine when the acceleration segment should be terminated. With the estimated friction parameter, one can calculate the future speed using (4) or (1) for the deceleration segment. While the values are guessed, the predicted speed curve may not be very precise. However, in our prediction procedure, we repredict the trajectory once a second. If the impact happens before the motor burns out, the inaccurate guess of the maximum speed and friction parameter will have no effects on the prediction accuracy. If the impact happens not too long (< 5 seconds) after burnout, the imprecision caused by the inaccurate guess of two values should be insignificant and acceptable. If the impact happens long after the burnout, the accuracy of the earlier prediction is not critical. Precise prediction can be obtained after the boosting period. Our prediction results presented in Section 4 confirm these discussions.

2.2 Estimation of Navigation Constant

Basic Estimation Approach

How a missile behaves relies on its control scheme. This study focuses on proportional navigation guidance because most missiles use the technique [3]. To briefly introduce the proportional navigation guidance technique, we can first consider the two-dimensional case. Figure 2 shows a two-dimensional missile-target engagement geometry. The line from the missile to the

target is called the line-of-sight (LOS). θ is the LOS angle and its time derivative is called the LOS rate. The LOS distance vector is denoted by \mathbf{R}_{TM} and its length is indicated by R_{TM} (a bold-face character for a vector and an Italic one for a scalar). V_T and V_M are the speeds of the target and the missile, respectively. A_c is the acceleration applied perpendicular to the line-of-sight. The proportional navigation control intends to make the missile heading rate (angular velocity) linearly proportional to the LOS rate. The purpose is to approximate a constant bearing course, that keeps the LOS angle constant. One proportional navigation law [3] is to issue acceleration commands, perpendicular to the instantaneous LOS, which are proportional to the LOS rate and closing speed. The law can be mathematically expressed as

$$A_c = \alpha V_c \omega_s \quad (6)$$

where α is the control gain known as the navigation constant, ω_s is the line-of-sight rate, and V_c is the closing speed which equals to $-R_{TM}(t)$.

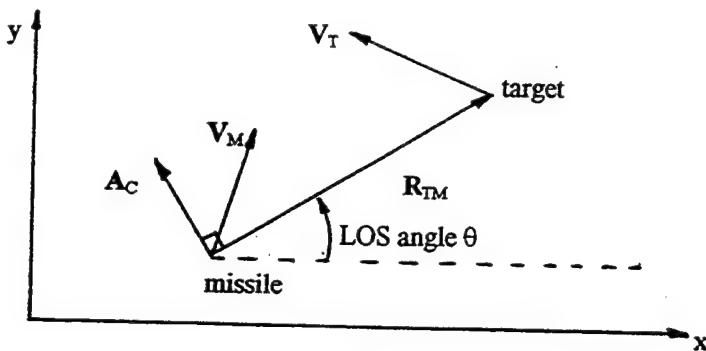


Figure 2 Two-dimensional missile-target engagement geometry

In control, (6) is used to determine the needed acceleration. But for estimation of α , A_c , V_c and ω_s are computed from the measured missile trajectory data, and α is computed from (6). This constant can be computed from $A_c(t)$, $V_c(t)$ and $\omega_s(t)$ at a single specific t . However, for the same

reason mentioned in estimating "fric", to reduce the effects of possible measurement errors, it is preferred to have α evaluated as an average of $A_c/(V_c\omega_s)$ from the data over a longer time period.

The real missile-target engagement is in three dimensional space. We need to deal with vectors. In the following, we will focus on computation of ω_s , A_c and V_c . Again, bold-faced symbols will be used to denote vectors. Given some trajectory data, with the use of the law of cosines, the line-of-sight rate can be estimated as (see Figure 3)

$$\omega_s(t) = \cos^{-1} \{ [(R_{TM})^2(t) + R_{TM}^2(t-\Delta t) - |R_{TM}(t) - R_{TM}(t-\Delta t)|^2] / [2R_{TM}(t)R_{TM}(t-\Delta t)] \} / \Delta t \quad (7)$$

where $|\cdot|$ indicates the vector length. Note that $R_{TM}(t)$, $R_{TM}(t-\Delta t)$ and $|R_{TM}(t) - R_{TM}(t-\Delta t)|$ are lengths of three sides of a triangle. The angular rate is about the axis perpendicular to both $R_{TM}(t-\Delta t)$ and $R_{TM}(t)$.

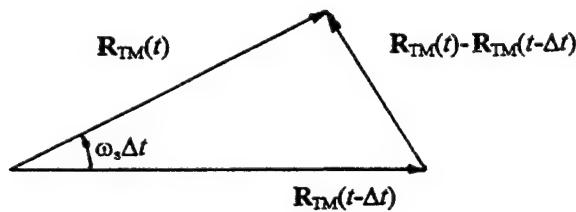


Figure 3. Relationship between ω_s and R_{TM} vectors

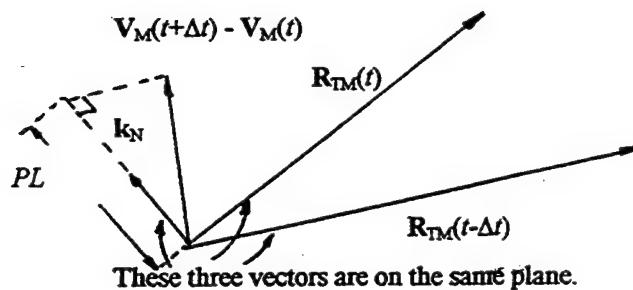


Figure 4. Illustration of vectors and the length used for computing $A_c(t)$.

Now let's turn our attention to A_c . The change of missile velocity direction is caused by the acceleration command A_c . The direction of $(\mathbf{V}_M(t+\Delta t) - \mathbf{V}_M(t))$, the same as that for $A_c(t)$, is supposed to be perpendicular to $\mathbf{R}_{TM}(t)$ and on the plane consisting of $\mathbf{R}_{TM}(t)$ and $\mathbf{R}_{TM}(t-\Delta t)$, i.e., along the \mathbf{k}_N axis as indicated in Figure 4. While the change of the measured velocity may not be on the direction of \mathbf{k}_N , for parameter estimation using this proposed technique, we project $\mathbf{V}_M(t+\Delta t) - \mathbf{V}_M(t)$ to the unit vector \mathbf{k}_N . The projection can be done by the following computations:

$$\mathbf{k}_N = \{[\mathbf{R}_{TM}(t-\Delta t) \times \mathbf{R}_{TM}(t)] \times \mathbf{R}_{TM}(t)\} / \|[\mathbf{R}_{TM}(t-\Delta t) \times \mathbf{R}_{TM}(t)] \times \mathbf{R}_{TM}(t)\| \quad (8)$$

$$\text{projected length } PL = \mathbf{k}_N \cdot (\mathbf{V}_M(t+\Delta t) - \mathbf{V}_M(t)) \quad (9)$$

The length is divided by Δt to give A_c for estimating the navigation constant, i.e.,

$$A_c(t) = PL/\Delta t \quad (10)$$

Another needed quantity for estimation is the closing velocity V_c , which is easy to obtain.

With a small effort in derivation, V_c can be obtained as

$$\begin{aligned} V_c(t) &= -\partial |\mathbf{R}_{TM}(t)| / \partial t \\ &= [\mathbf{R}_{TM}(t) \cdot \mathbf{V}_{TM}(t)] / |\mathbf{R}_{TM}(t)|. \end{aligned} \quad (11)$$

With $A_c(t)$, $V_c(t)$ and $\omega_s(t)$ computed from the missile and target trajectory data, the navigation constant can be solved from (6).

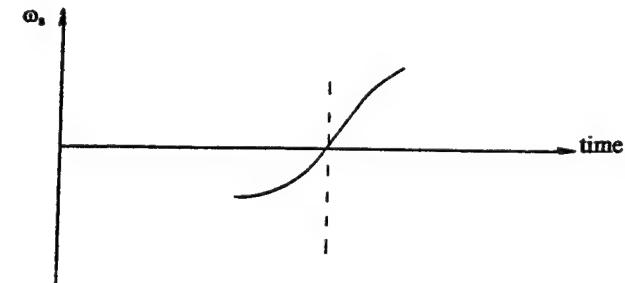
Weighted Average of the Estimated Navigation Constant Values

The navigation constant estimated using data at t_i is denoted by $\hat{\alpha}_i$, which is computed as

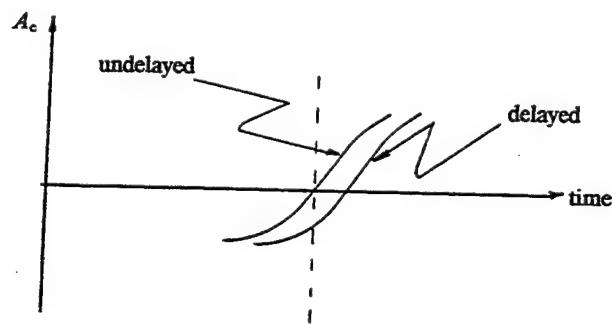
$$\hat{\alpha}_i = A_c(t_i) / V_c(t_i) \omega_s(t_i) \quad (12)$$

Estimation at some time instants could be unreliable. In (12), if ω_s is small, a small error on it can cause a large error on $\hat{\alpha}_i$. Figure 5 illustrates one possible problem caused by a small ω_s . With mechanical parts involved in missile control, the time delay on control response can be significant. A commanded acceleration A_c may be observed after a delayed time period. Consequently, when we use (12) to compute $\hat{\alpha}_i$, the lagged A_c is used and it introduces error. With a small ω_s , the error is amplified. Figure 5 shows that ω_s changes sign and A_c is lagged. Near the zero crossing point, the

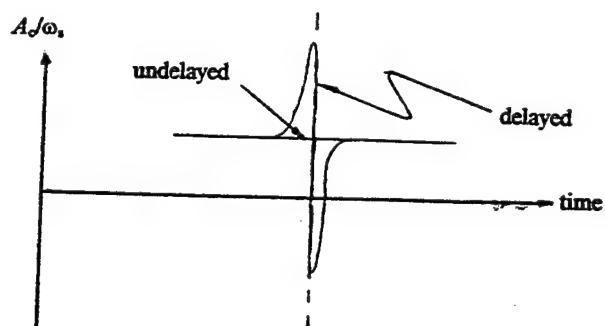
magnitude of the estimated $\hat{\alpha}_i$ is unreasonably large. Figure 6 provides the plots of ω_s , $\hat{\alpha}_i$ and $\hat{\alpha}$ ($\hat{\alpha}$ will be explained later) obtained for one missile simulation run (M1RUN2). The plot of $\hat{\alpha}_i$ in (b) does show peaks when ω_s is small.



(a) ω_s

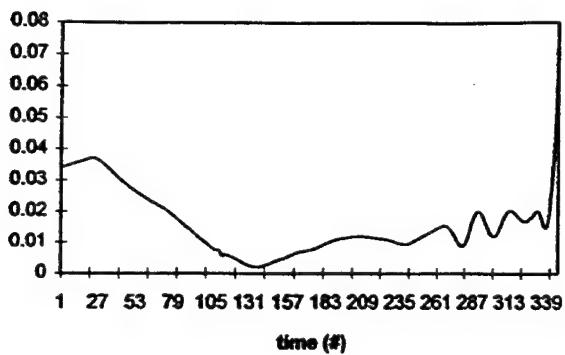


(b) A_c and delayed A_c

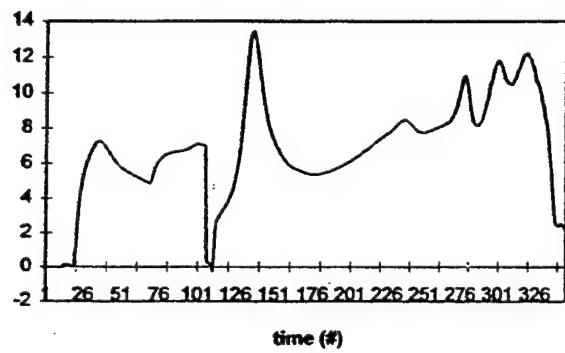


(c) A_c/ω_s

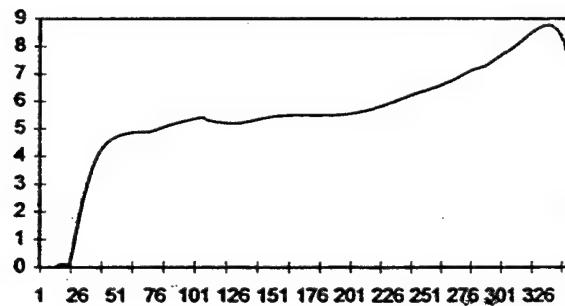
Figure 5. Explanation for the cause of estimation error on α



(a) ω_s for M1RUN2



(b) $\hat{\alpha}_i$ for M1RUN2



(c) $\hat{\alpha}$ for M1RUN2

Figure 6. Effects of a small ω_s observed in one simulation run

It has been seen that ω_s is an important factor in determining the reliability of an estimation. A larger one gives a more reliable estimation. Thus it is reasonable to use ω_s as a weighting for computing the weighted average of $\hat{\alpha}_i$'s, as an estimate of α . The weight is defined as

$$W_i = \begin{cases} 0 & \text{if } \hat{\alpha}_i < 0 \\ \omega_s & \text{if } 0 < \hat{\alpha}_i < 10 \\ 10\omega_s / \hat{\alpha}_i & \text{if } \hat{\alpha}_i > 10 \end{cases} \quad (13)$$

The weight is assigned zero when the estimated navigation constant is negative (line 1) and assigned a value smaller than ω_s if the estimated α_i is too large (line 3). If $\hat{\alpha}_i$ is between 0 and 10, the weight is set equal to ω_s (line 2). A smaller ω_s gives less reliable estimation and thus the corresponding $\hat{\alpha}_i$ is assigned a smaller weight.

With the weight defined, α is estimated as

$$\hat{\alpha} = \begin{cases} \min \left(\frac{\sum_i \beta^{i-1} W_i \hat{\alpha}_i}{\sum_i \beta^{i-1} W_i}, 30 \right), & \text{or} \\ 10 & \text{if } \sum_i \beta^{i-1} W_i \hat{\alpha}_i = 0 \left(\text{no reliable } \hat{\alpha}_i \text{ yet} \right) \end{cases} \quad (14)$$

The first part of (14) computes the weighted average and has the value limited under 30. This limitation is occasionally activated at the early stage of prediction while very few reliable $\hat{\alpha}_i$'s are available. The second part simply sets α to 10 if all $\hat{\alpha}_i$'s are non-positive. This situation is also very unusual and can happen only when no valid estimation is available at the beginning.

3. Prediction Procedure

The trajectory prediction starts shortly after the missile is launched. Figure 7 shows the flow chart of the prediction procedure. The procedure uses a sliding sample window for continually updating prediction so that the warning information can be updated and be more accurate and reliable

while the missile is getting closer to its target. The prediction consists of two major parts; one is the prediction made while the motor is still burning and the other is the prediction made after burnout. The prediction made during the boosting utilizes past speed data to estimate the acceleration, the assumed maximum speed to switch from the acceleration segment into the deceleration segment in prediction, and the assumed friction parameter value to predict the speed in the deceleration period. Upon burnout, the trajectory data is used to estimate the friction parameter for predicting the future speed. In both situations, the navigation constant is estimated using (14).

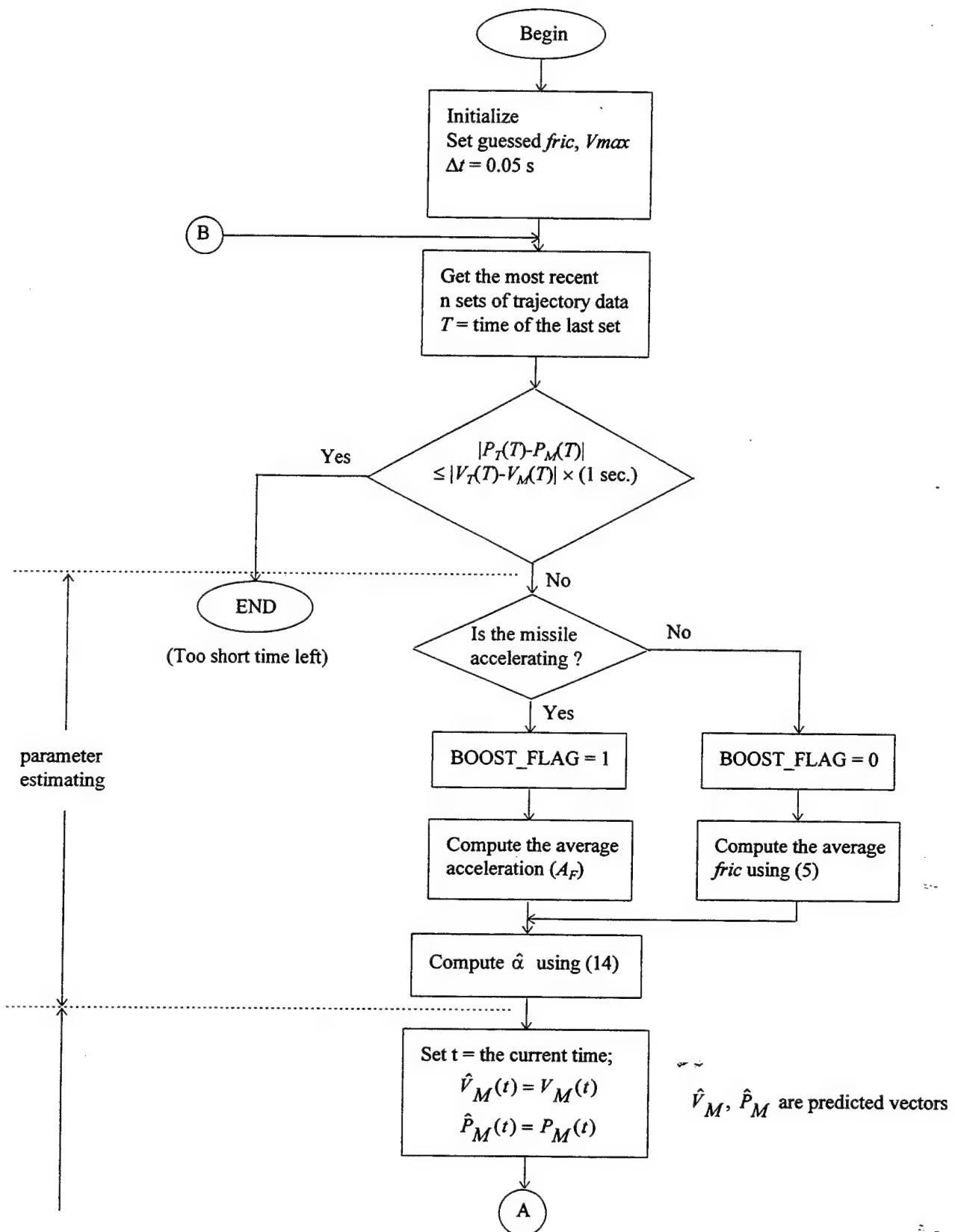


Figure 7. The prediction procedure (to be continued)

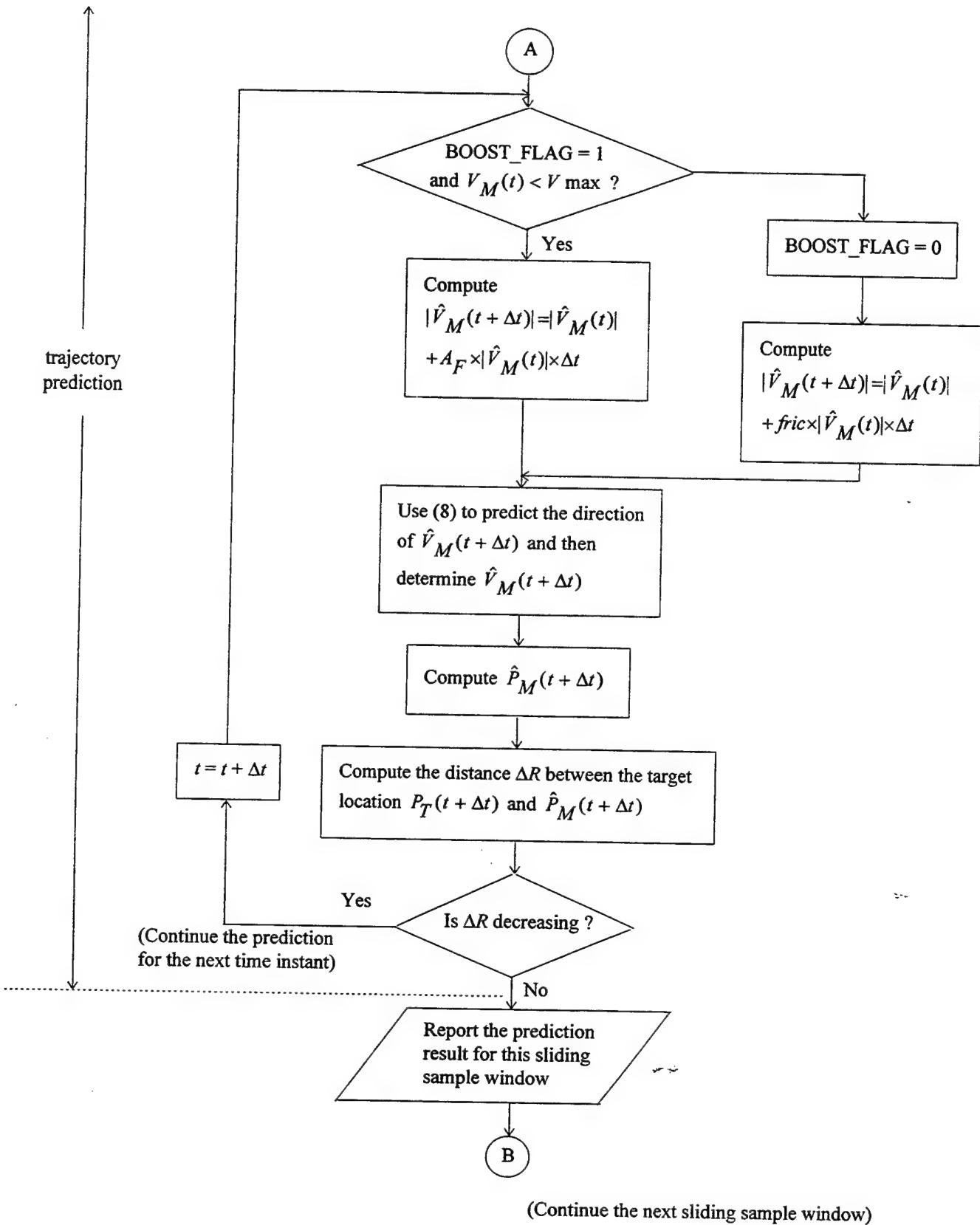


Figure 7. The prediction procedure

4. Experimental Results

4.1 Trajectory Data for Experiments

Twenty seven sets of missile flyout data were provided by Wright Laboratory for usage in this study. Those data were obtained from simulations for three different types of missiles. The twenty seven sets of data are named M1RUN1, M1RUN2, ... M3RUN9, of which each specifies the missile type and the simulation run. Information about the time of impact and maneuvering for these simulation runs are listed in Table 1.

Table 1. Information for used simulation runs

Run Name	Time of Impact	Special Maneuvering
M1RUN1	Prior to burnout	No
M1RUN2	Longer than 10 seconds after burnout	No
M1RUN3	Longer than 10 seconds after burnout	No
M1RUN4	Prior to burnout	No
M1RUN5	Prior to burnout	No
M1RUN6	Prior to burnout	Yes
M1RUN7	0.7 seconds after burnout	Yes
M1RUN8	2.7 seconds after burnout	No
M1RUN9	4.6 seconds after burnout	No
M2RUN1	Longer than 10 seconds after burnout	No
M2RUN2	Longer than 10 seconds after burnout	No
M2RUN3	Longer than 10 seconds after burnout	No
M2RUN4	Prior to burnout	No
M2RUN5	Prior to burnout	No
M2RUN6	2.1 seconds after burnout	Yes
M2RUN7	3.7 seconds after burnout	Yes
M2RUN8	3.0 seconds after burnout	No
M2RUN9	1.3 seconds after burnout	No
M3RUN1	Longer than 10 seconds after burnout	No
M3RUN2	Longer than 10 seconds after burnout	No
M3RUN3	Longer than 10 seconds after burnout	No
M3RUN4	Prior to burnout	No
M3RUN5	0.7 seconds after burnout	No
M3RUN6	7.5 seconds after burnout	Yes
M3RUN7	9.4 seconds after burnout	Yes
M3RUN8	4.3 seconds after burnout	No
M3RUN9	5.6 seconds after burnout	No

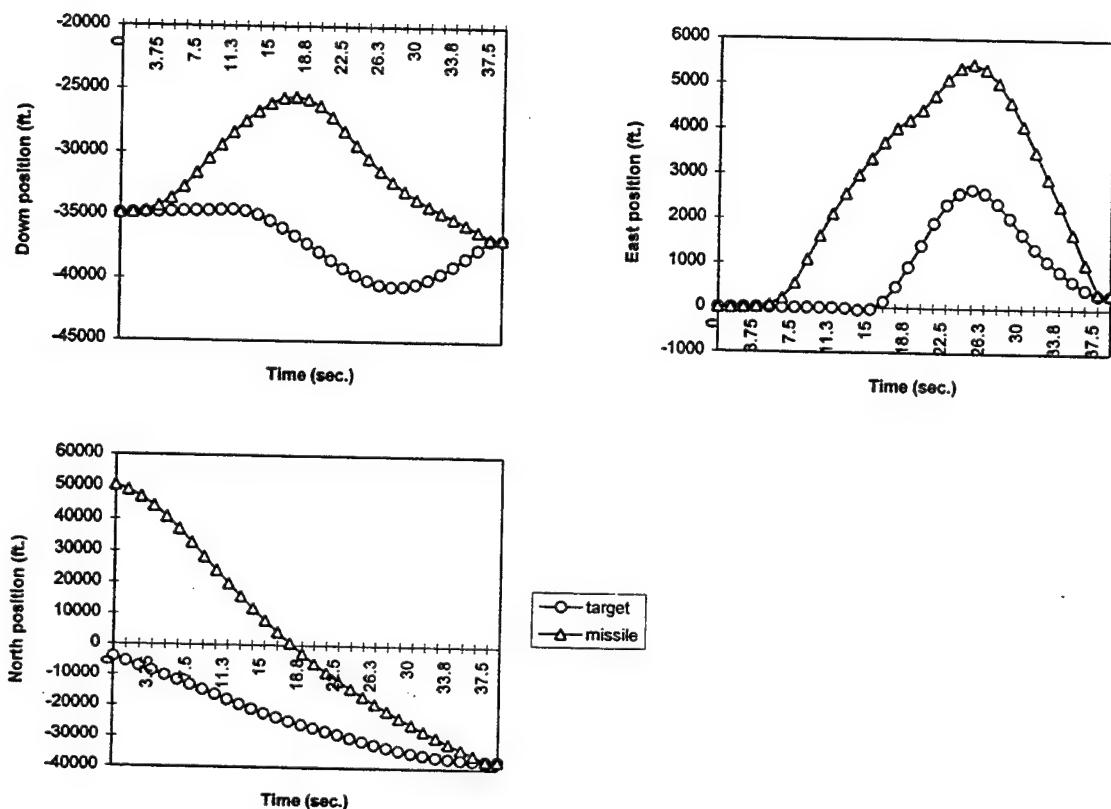


Figure 8. Trajectory plots for M3RUN3

Each data file consists of the velocities and positions of the target and the missile for the complete period from launching to impact. Velocities and positions are represented in three coordinates: North, East and Down. Observations from the data indicate that the motor burning periods for the three types of missiles are about 5.4 seconds, 10 seconds and 7.5 seconds respectively. The maximum velocities are 3,700, 4,400 and 3,800 ft/sec. Simulation runs 1 to 3 for all three types of missiles have the altitude at near 35,000 feet. Other runs have the altitude at about 9,000 feet. Due to the difference in altitude, the friction in runs 4 to 9 is larger and the maximum velocities are lower. Trajectories for the first two types are pretty normal while those for the third one look trickier. Figure 8 shows the missile and target trajectories in data M3RUN3, in which the missile tries to turn into east away from the target at the early stage. There may be an offset added on

purpose to the actual target position in control. This may be done just to create the confusion on which object is the real target. In this example, the offset could be about 4,000 feet in the east direction and 10,000 feet up in the altitude. However, in the last 12-15 seconds, the offset seems removed and the missile turns back toward the target.

4.2 Experiment Setup and Results

In the experiments, the prediction starts 2 seconds after the missile is launched. The prediction is repeated every second using new data. Two cases are tested; one is that the missile type is unknown and the other is that the missile type has been identified. As mentioned previously, one group of simulation runs has the altitude near 35,000 feet and the other has the altitude near 9,000 feet. Thus different preset friction parameter and maximum speed values are used. While the missile type is unknown, the maximum missile speed is set to 3,996 ft/sec at the altitude near 35,000 feet, and 3222 ft/sec for the altitude near 9,000 feet. The friction parameter is set to -0.03613 and -0.08191 for the altitude near 35,000 and 9,000 feet, respectively. While the missile type is known, the following data are used:

- Missile 1: Maximum speed = 3652, $fric = -0.05444$ when the altitude is near 35,000 feet
Maximum speed = 2938, $fric = -0.14310$ when the altitude is near 9,000 feet
- Missile 2: Maximum speed = 4550, $fric = -0.03424$ when the altitude is near 35,000 feet
Maximum speed = 3351, $fric = -0.07081$ when the altitude is near 9,000 feet
- Missile 3: Maximum speed = 3671, $fric = -0.02581$ when the altitude is near 35,000 feet
Maximum speed = 3290, $fric = -0.06243$ when the altitude is near 9,000 feet

Trajectories and prediction errors have been obtained and plotted.

Trajectory Curves

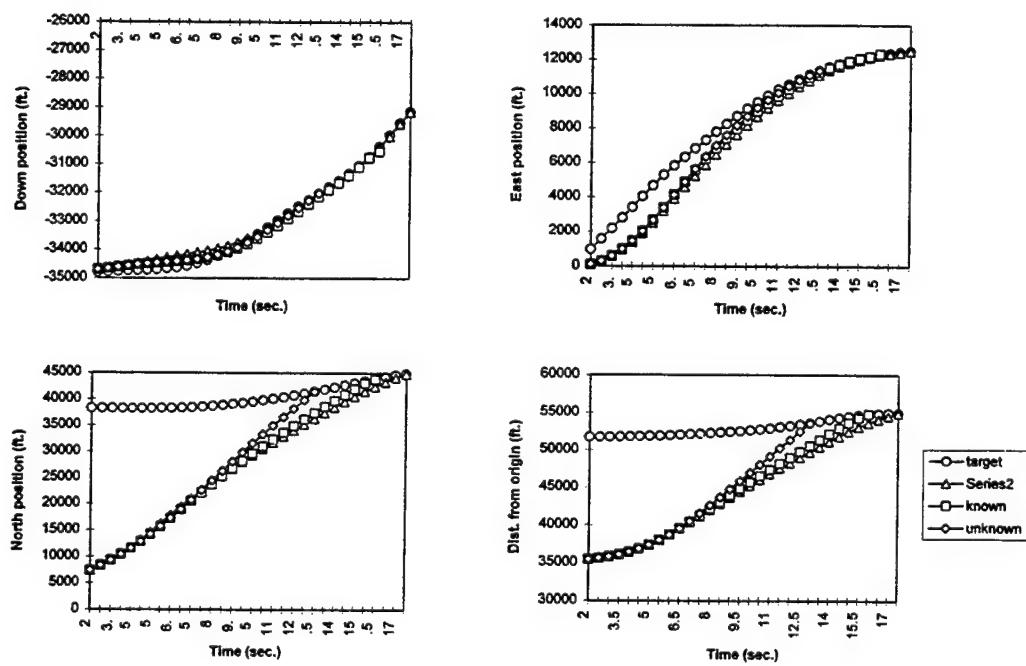
Selected prediction results for twelve cases are provided in Figure 9. In each plot, there are four curves showing the target trajectory, the real missile trajectory and two predicted missile

trajectories (one with the missile type known and another unknown). The fourth figure for each run shows the distances from the origin to the target, to the missile, and to two predicted missile locations. All predictions were made at the time t equal to 2 seconds. While most predictions were made long before impact, errors could be large. However, the errors will reduce dramatically while the missile is getting closer to its target and the prediction is redone. This will be seen later in Figure 10.

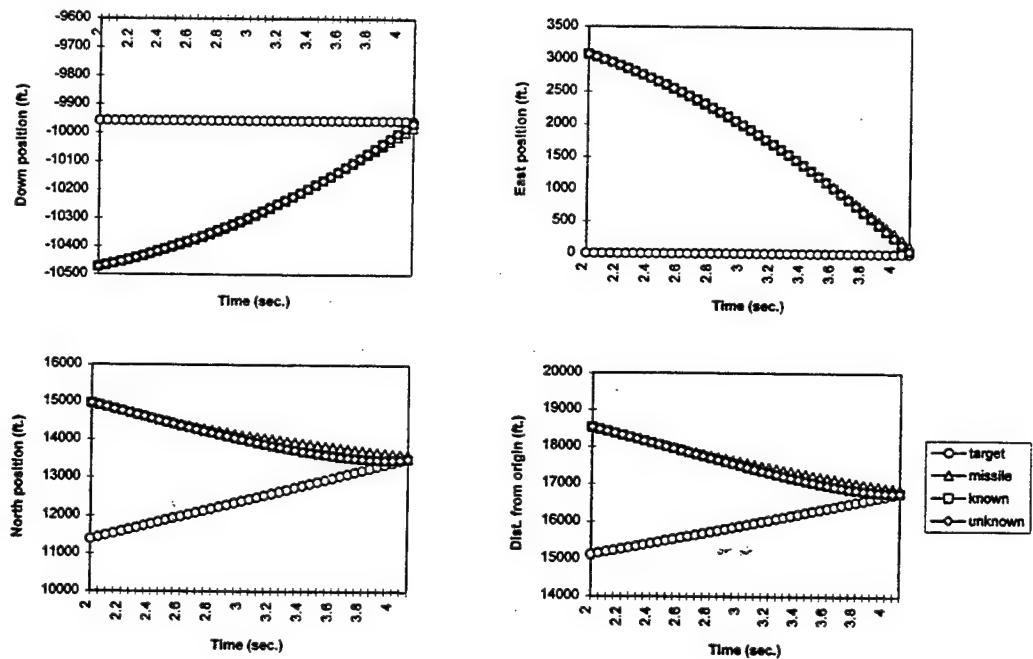
Errors on Time and Location of Impact

Knowing when the impact will occur can be important. An airplane can sharply turn away from the missile-target plane or employ a countermeasure shortly before the impact to efficiently escape from the attack [4].

In our experiments, prediction was made every second. For each prediction, errors on location and time of impact were evaluated. The errors are plotted in Figure 10. A negative error on impact time denotes that the predicted impact occurs earlier than the real one. The error on location is plotted using the distance between the actual impact location and the predicted impact location. The errors usually drop while the time left before the impact is getting shorter. Table 2 shows the position vector from the target to the missile at the real impact, the position vector from the target to the predicted missile location at the predicted time of impact and the error on the time of impact. Predictions were made 4 seconds and 6 seconds before impact. The missile type was assumed unknown. For the case that the predicted impact happens after the real impact, prediction cannot be continued because the flyout data file does not include the expected target trajectory upon impact. Thus for those cases, the time error was estimated as the distance between the missile and the target divided by the relative velocity between them. For these situations, the predicted position vectors in the table are the vectors at the time of real impact, not the predicted time of impact. They are marked by **. The results with the missile type known should be much better.



(a1) Trajector plots for M1RUN2



(a2) Trajector plots for MIRUN4

Figure 9. Trajectory plots (to be continued)

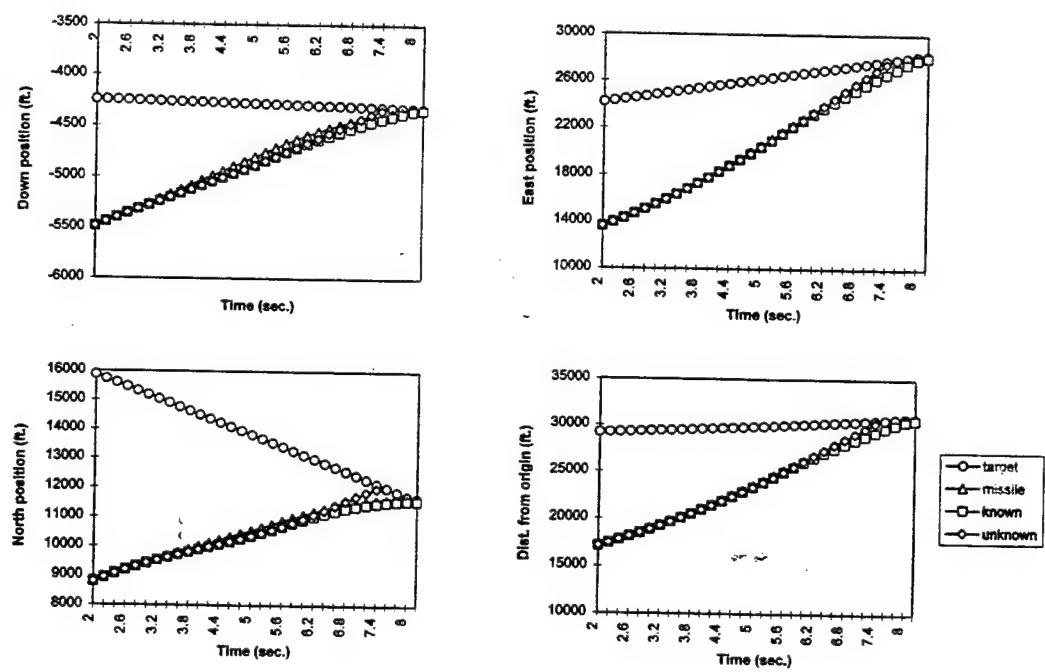
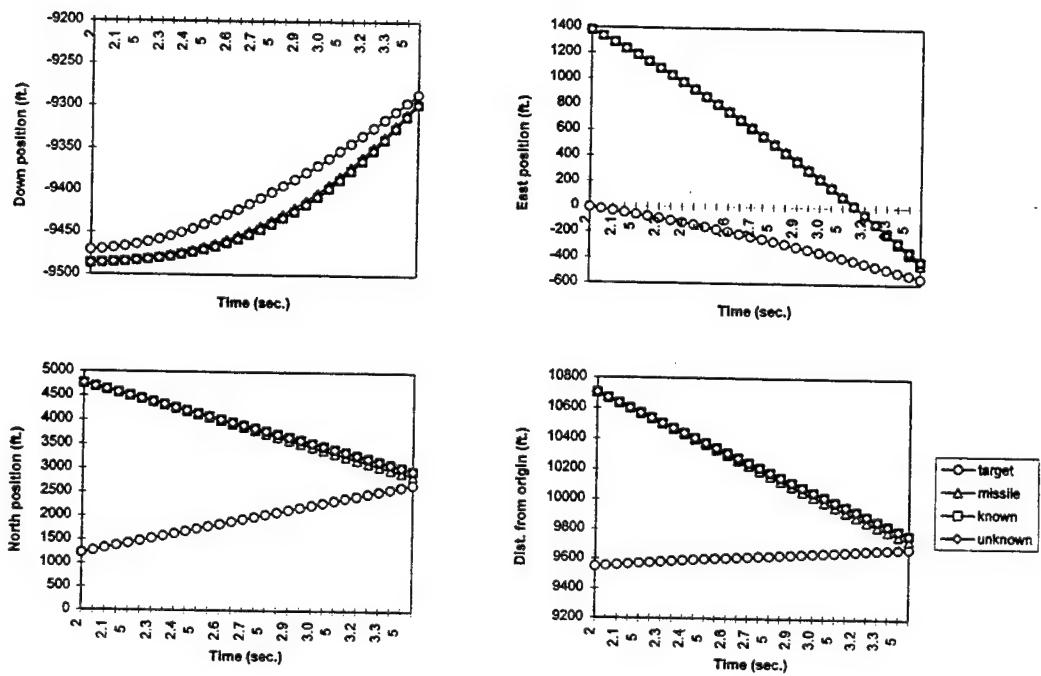


Figure 9. Trajectory plots (to be continued)

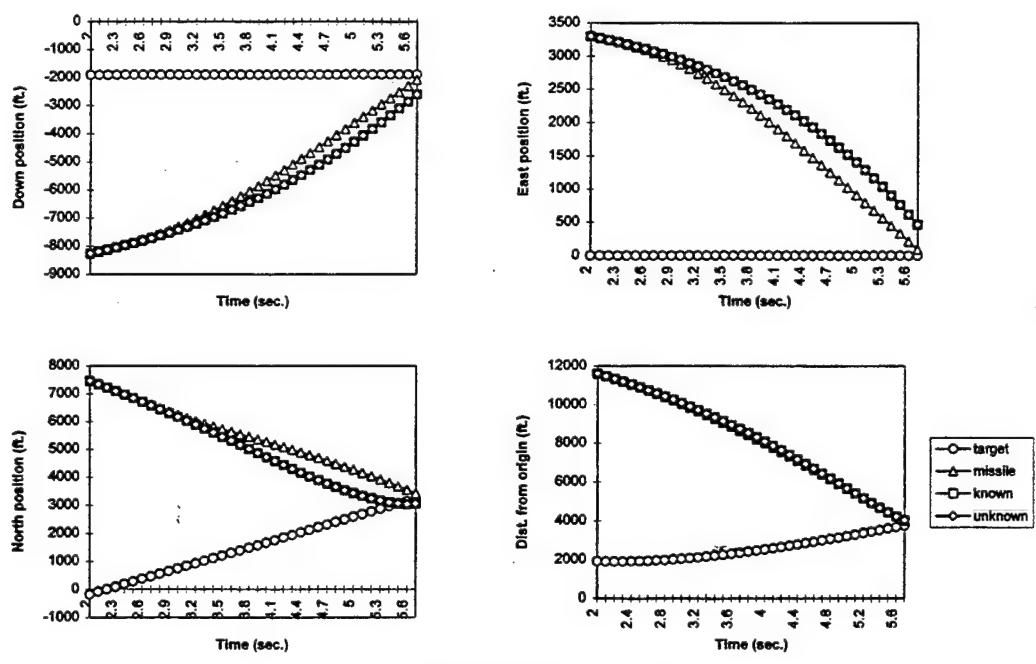
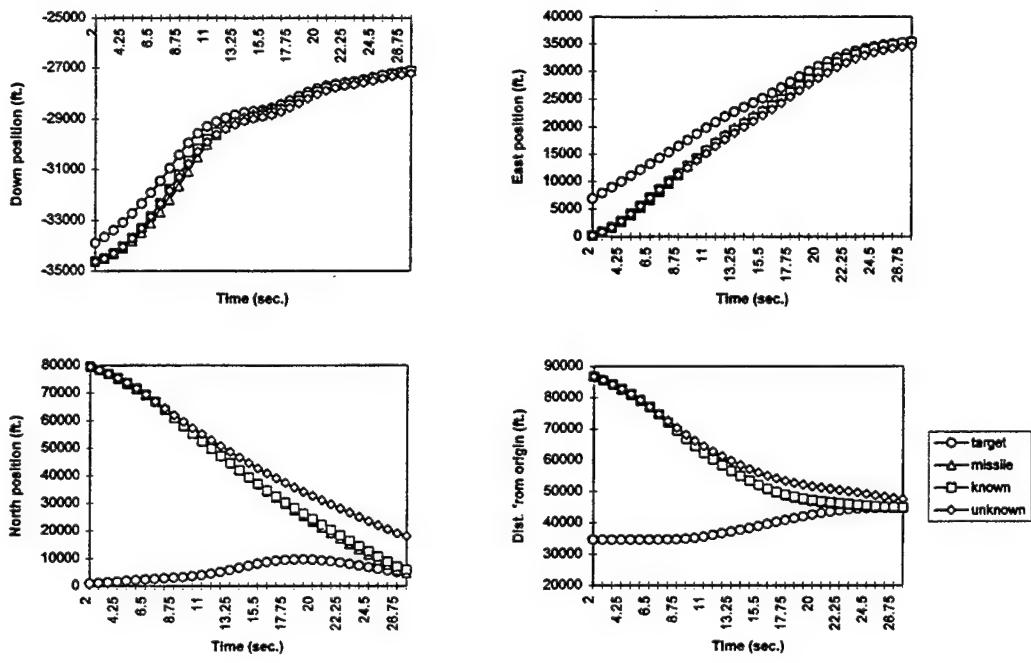
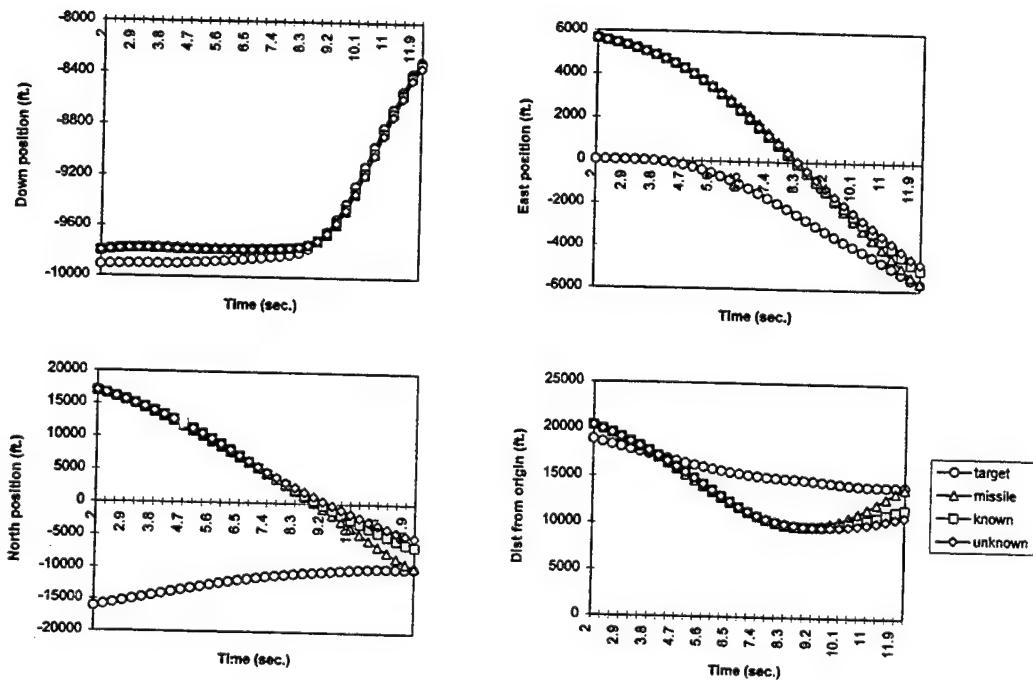
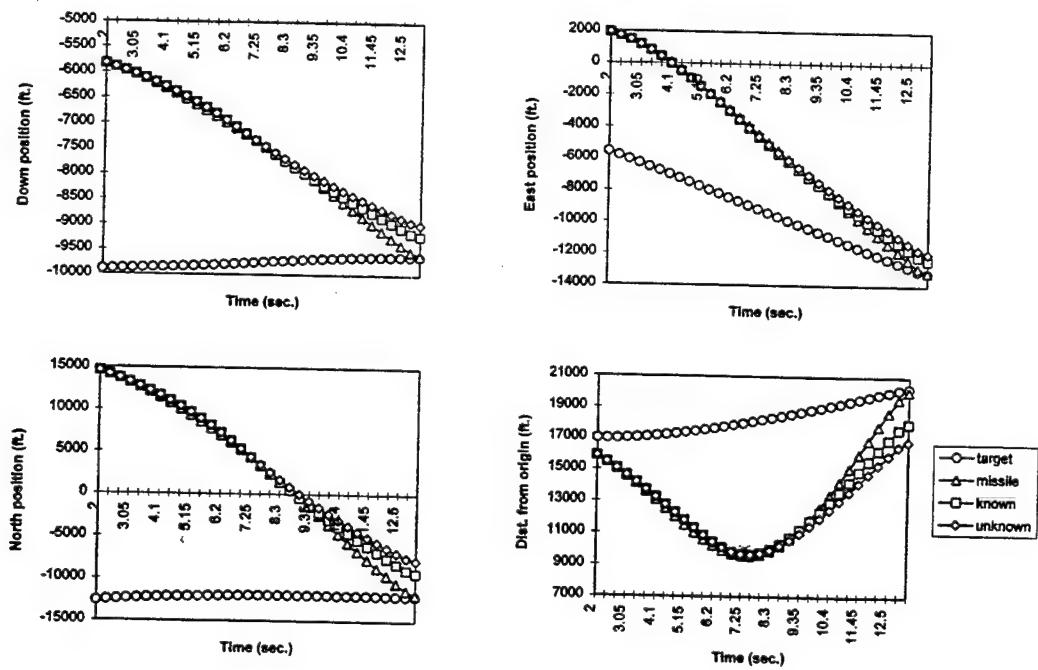


Figure 9. Trajectory plots (to be continued)

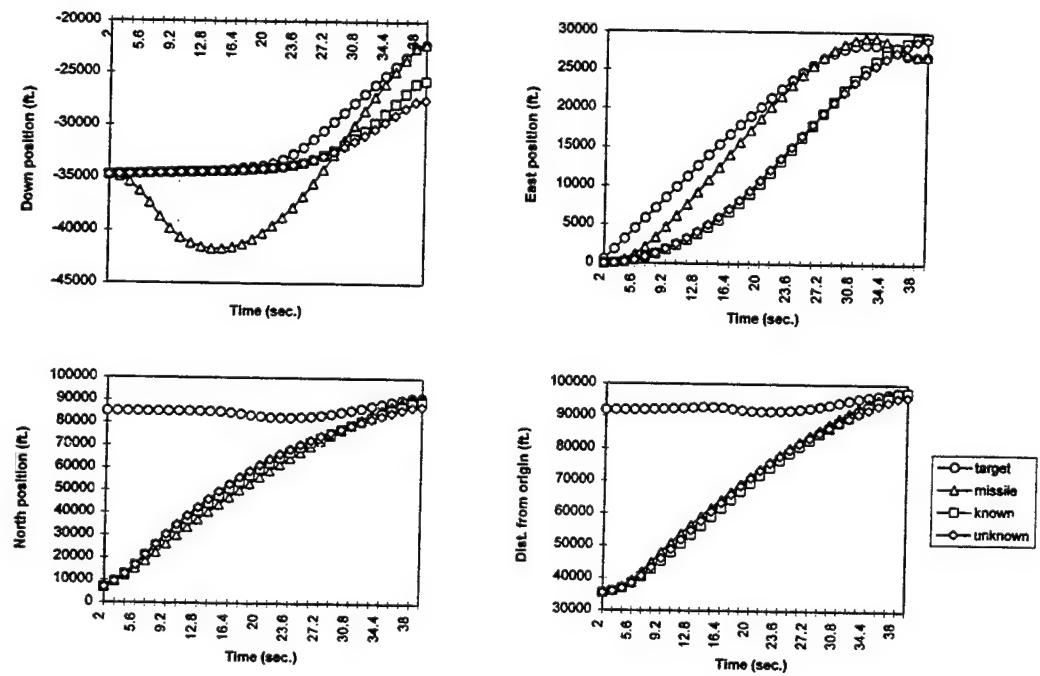


(b3) Trajector plots for M2RUN6

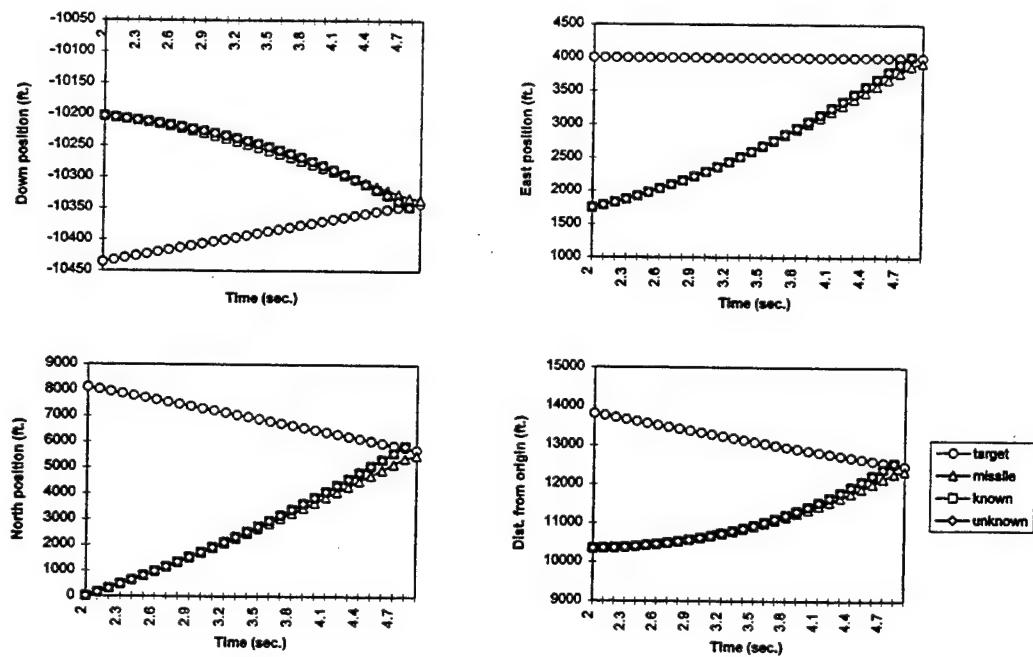


(b4) Trajector plots for M2RUN8

Figure 9. Trajectory plots (to be continued)

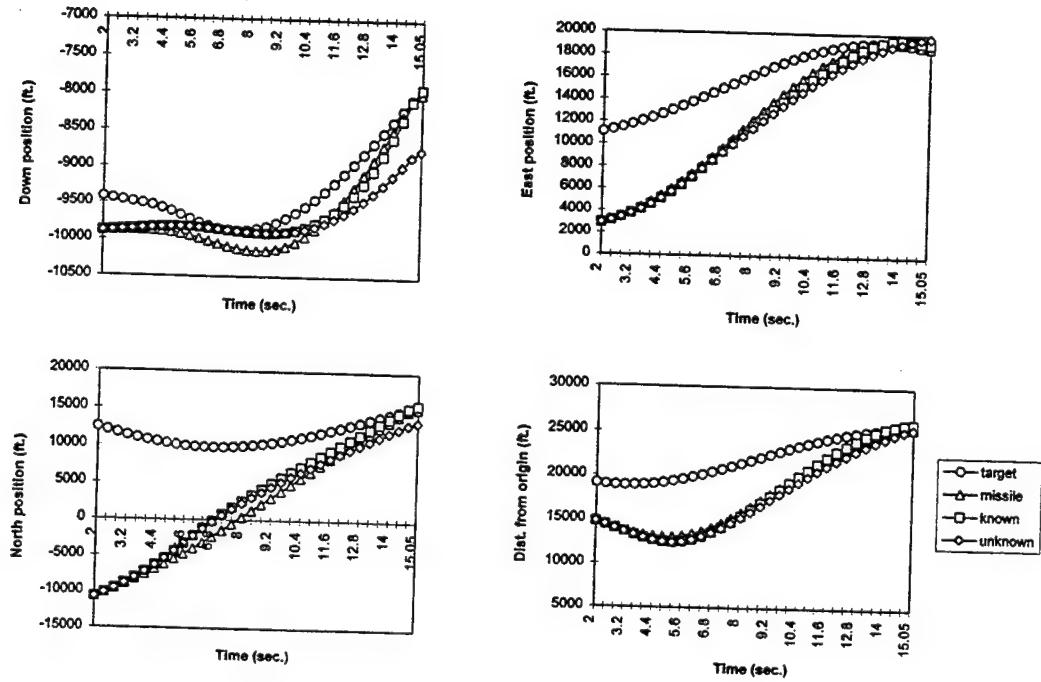


(c1) Trajector plots for M3RUN2

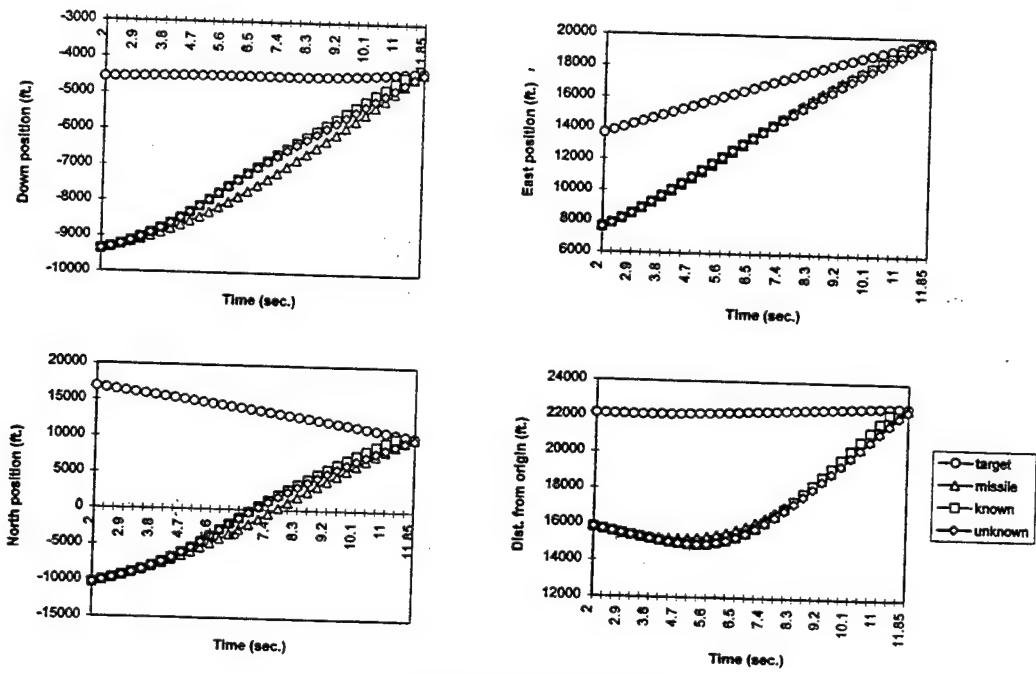


(c2) Trajector plots for M3RUN4

Figure 9. Trajectory plots (to be continued)

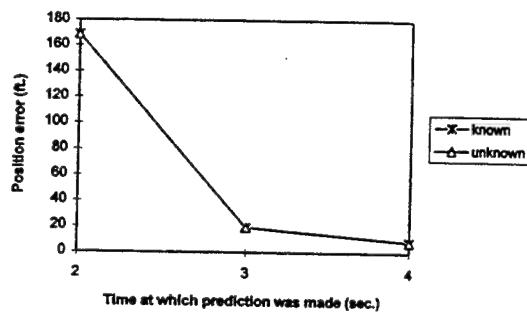
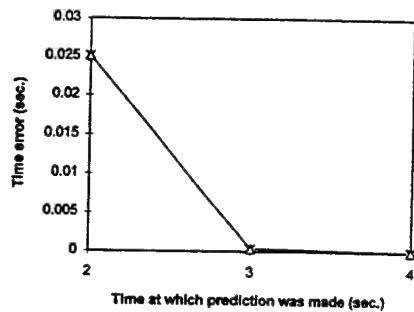


(c3) Trajector plots for M3RUN6

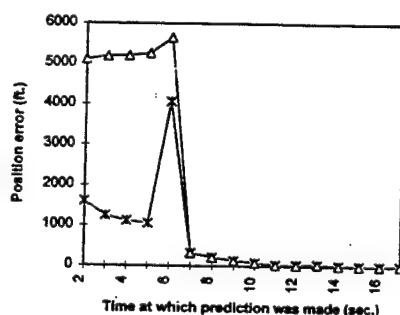
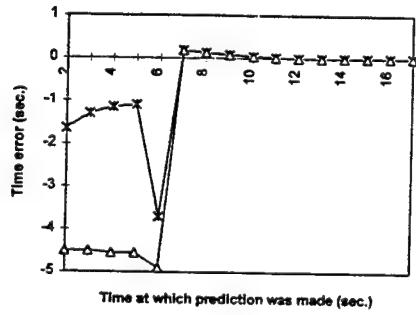


(c4) Trajector plots for M3RUN8

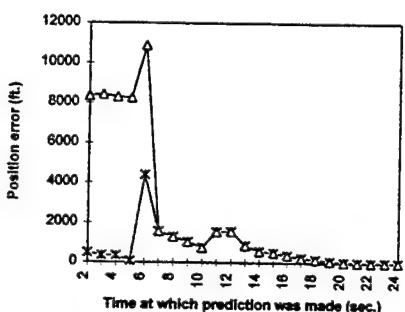
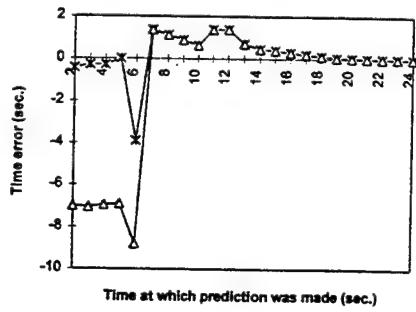
Figure 9. Trajectory plots



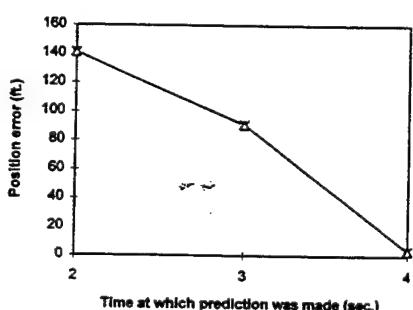
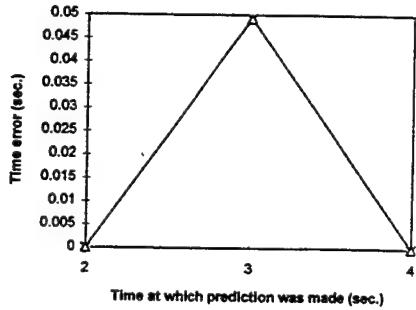
(a1) Prediction errors for M1RUN1



(a2) Prediction errors for M1RUN2

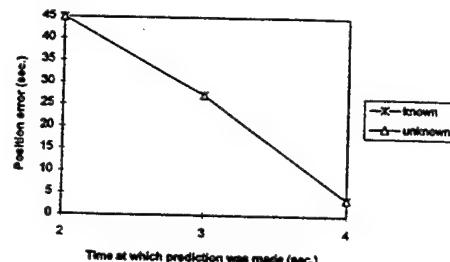
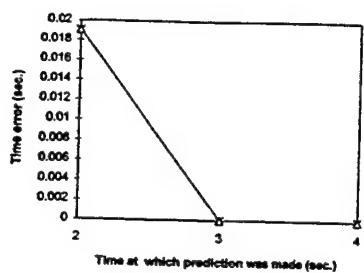


(a3) Prediction errors for M1RUN3

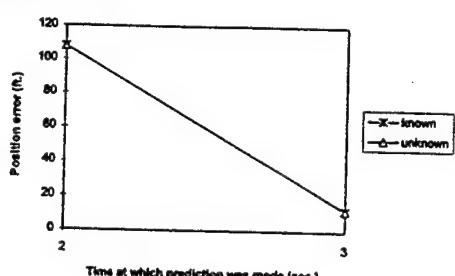
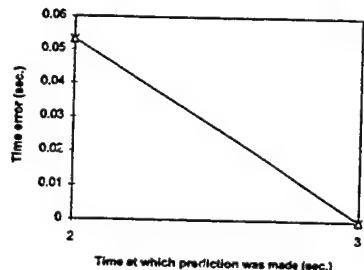


(a4) Prediction errors for M1RUN4

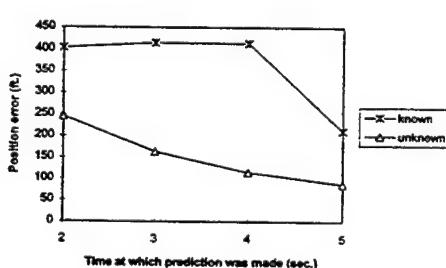
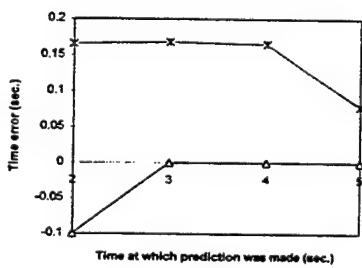
Figure 10. Prediction errors (to be continued)



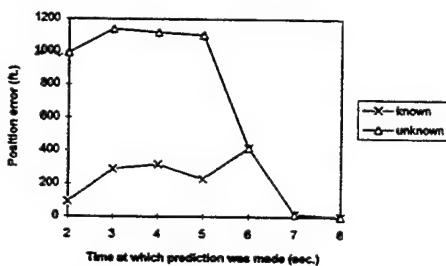
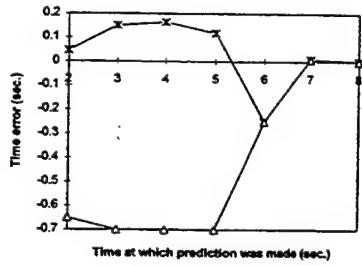
(a5) Prediction errors for MIRUN5



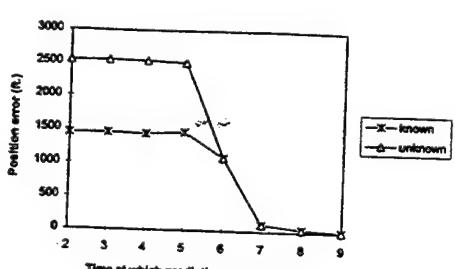
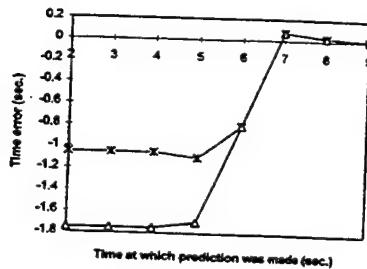
(a6) Prediction errors for MIRUN6



(a7) Prediction errors for MIRUN7

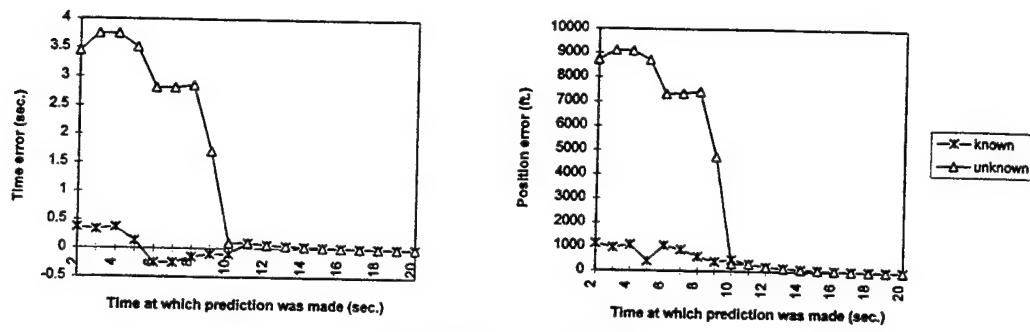


(a8) Prediction errors for MIRUN8

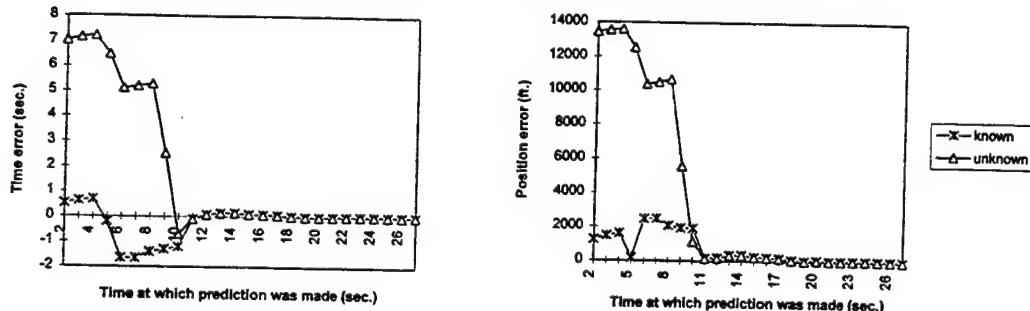


(a9) Prediction errors for MIRUN9

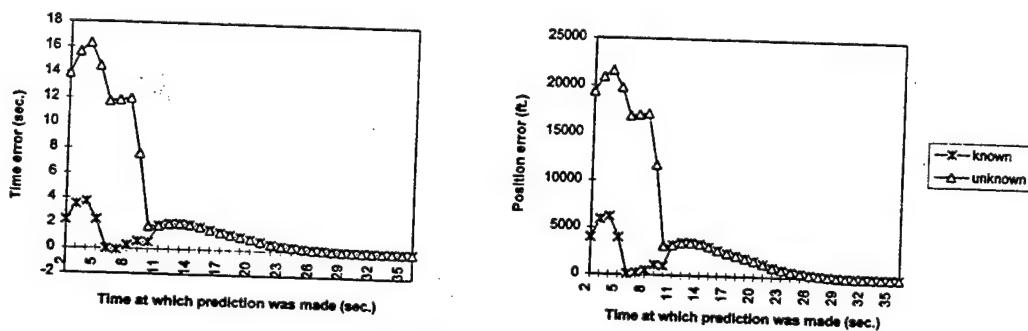
Figure 10. Prediction errors (to be continued)



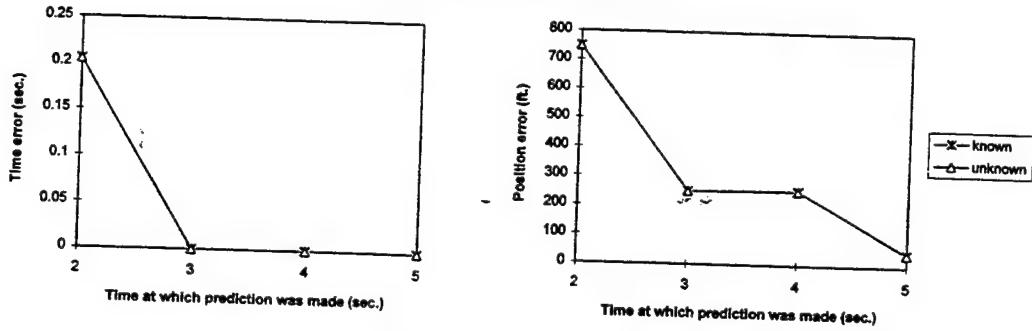
(b1) Prediction errors for M2RUN1



(b2) Prediction errors for M2RUN2

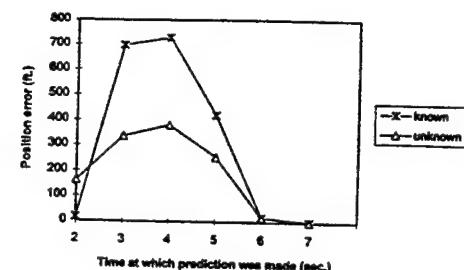
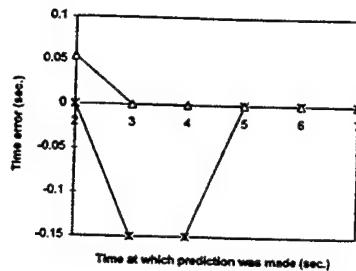


(b3) Prediction errors for M2RUN3

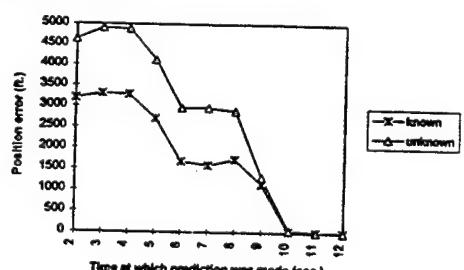
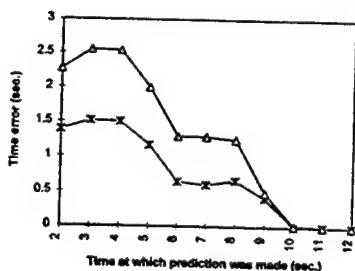


(b4) Prediction errors for M2RUN4

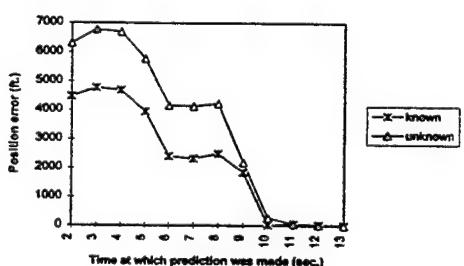
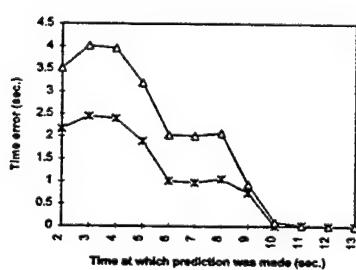
Figure 10. Prediction errors (to be continued)



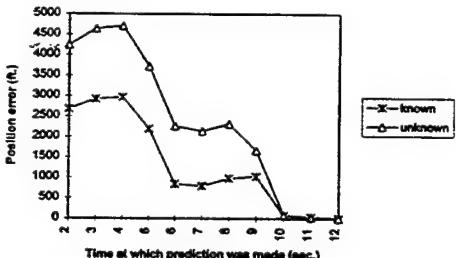
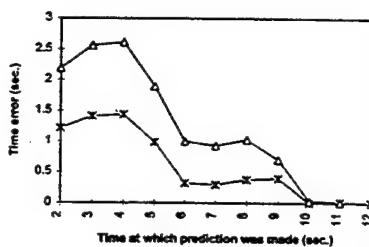
(b5) Prediction errors for M2RUNS



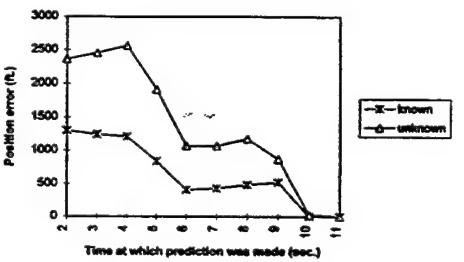
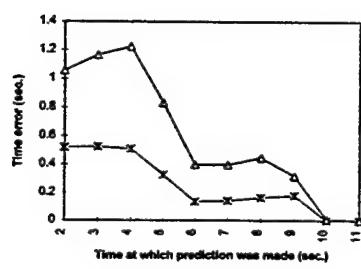
(b6) Prediction errors for M2RUN6



(b7) Prediction errors for M2RUN7

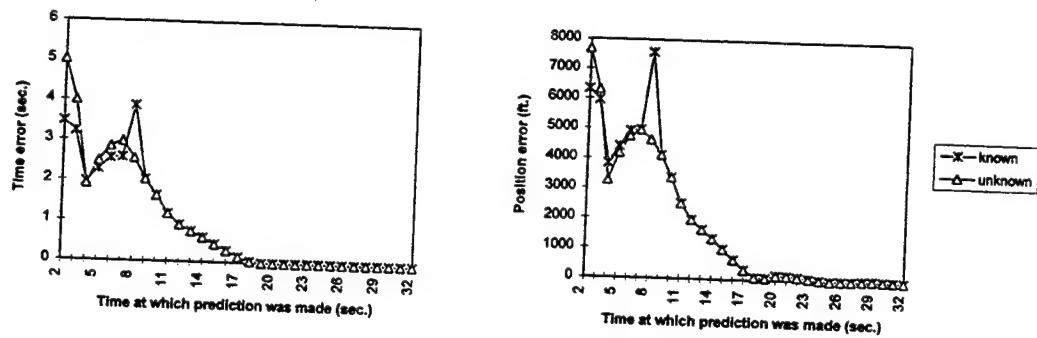


(b8) Prediction errors for M2RUN8

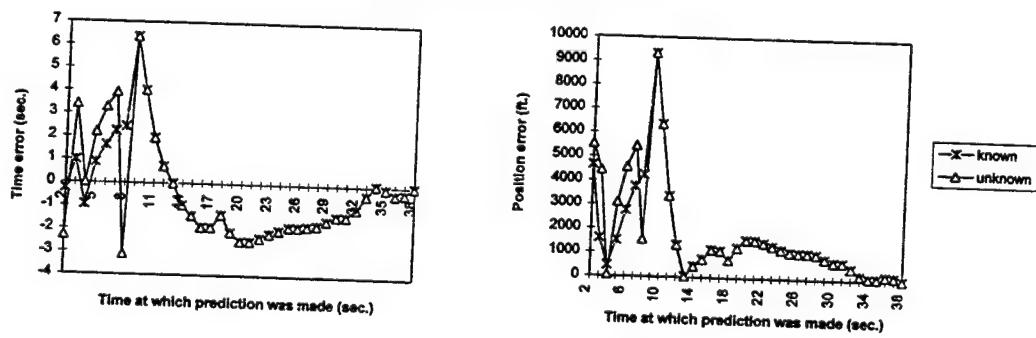


(b9) Prediction errors for M2RUN9

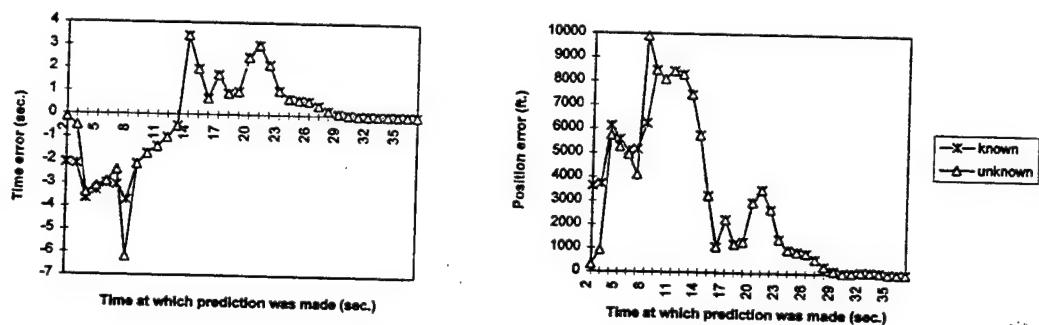
Figure 10. Prediction errors (to be continued)



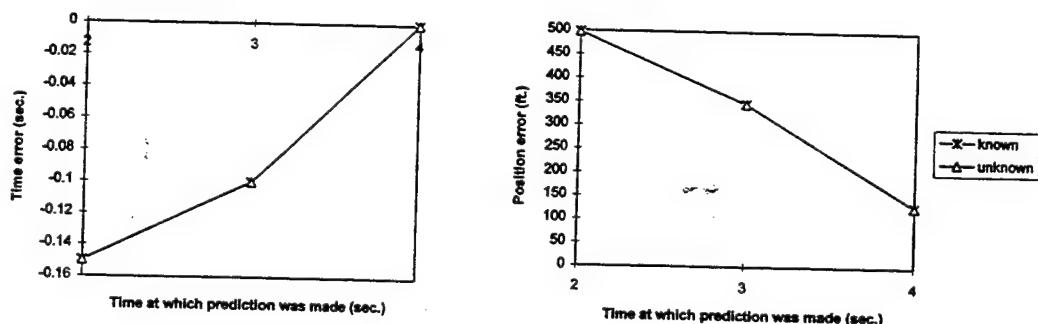
(c1) Prediction errors for M3RUN1



(c2) Prediction errors for M3RUN2



(c3) Prediction errors for M3RUN3



(c4) Prediction errors for M3RUN4

Figure 10. Prediction errors (to be continued)

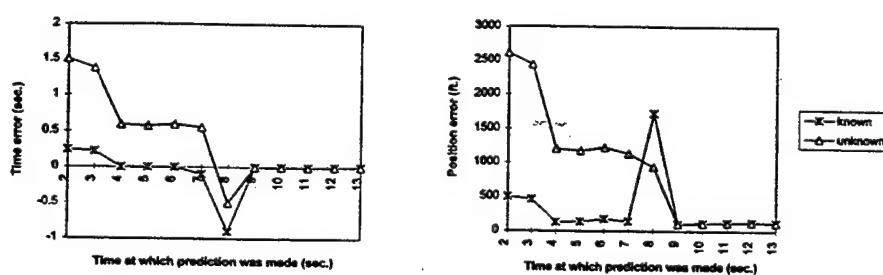
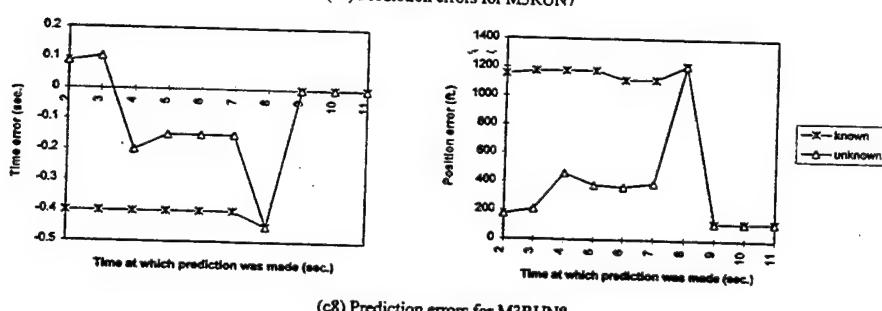
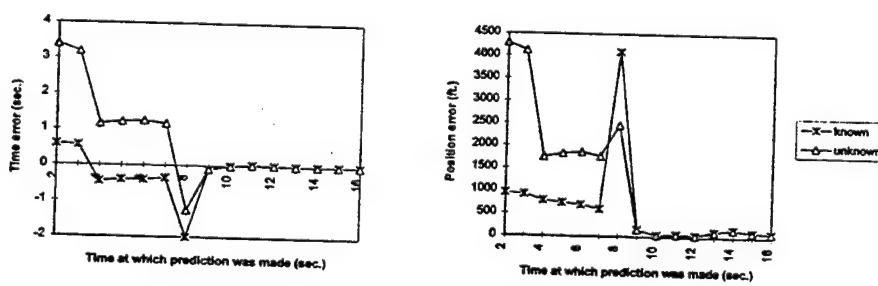
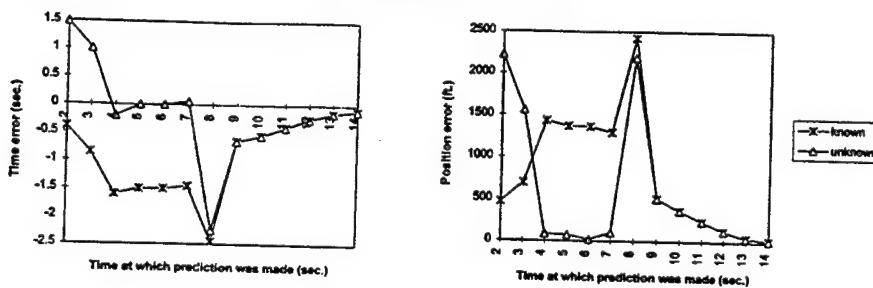
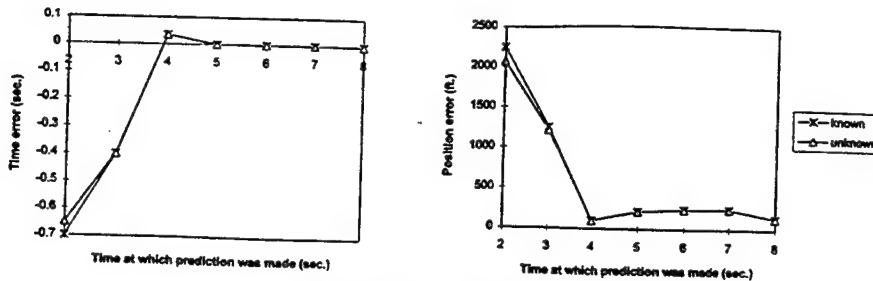


Figure 10. Prediction errors

Table 2. Data for prediction made about 4 and 6 seconds before the real impact

Run name	Prediction at 5.5 to 6.5 seconds prior to impact			Prediction at 3.5-4.5 seconds prior to impact (First prediction if the time is shorter than 5.5 seconds)		
	Actual position vector	Predicted position vector	Error on impact time	Actual position vector	Predicted position vector	Error on impact time
MIRUN1				0,-322,13	0,-365,176	≈0 sec.
MIRUN2	-3,81,-6	0,-53,-7	≈0 sec.	-3,81,-6	-1,-64,-7	≈0 sec.
MIRUN3	1,79,11	**2,163,17	≈0.50 sec.	1,79,11	2,101,12	≈0 sec.
MIRUN4				134,108,-22	46,-1,8	≈0 sec.
MIRUNS				-76,-149,19	-98,-189,22	≈0 sec.
MIRUN6				110,189,-9	**133,294,12	≈0.05 sec.
MIRUN7				103,-92,-5	4,-2,0	≈-0.10 sec.
MIRUN8	-81,42,5	-38,-23,-4	≈-0.65 sec.	-81,-42,-5	-3,-1,0	≈-0.70 sec.
MIRUN9	39,68,25	-1,-3,-1	≈-1.75 sec.	39,68,25	-10,-22,-7	≈-0.80 sec.
M2RUN1	-4,-206,2	-5,303,1	≈0 sec.	-4,-206,2	-4,251,1	≈0 sec.
M2RUN2	-4,119,0	-2,90,-1	≈0 sec.	-4,119,0	-5,160,-1	≈0 sec.
M2RUN3	-4,146,4	-6,156,3	≈0 sec.	-4,146,4	-4,147,3	≈0 sec.
M2RUN4				95,178,-181	**468,-194,-711	≈0.20 sec.
M2RUN5				-3,348,-16	**4,508,-22	≈0.05 sec.
M2RUN6	49,258,-3	**643,3164,45	≈1.30 sec.	49,258,-3	635,3086,-45	≈1.25 sec.
M2RUN7	23,189,5	**551,4291,91	≈2.00 sec.	22,189,5	**309,2366,58	≈0.95 sec.
M2RUN8	59,199,30	**661,2227,334	≈0.90 sec.	59,199,30	**530,1779,268	≈0.70 sec.
M2RUN9	20,245,38	**172,2132,284	≈0.80 sec.	20,245,38	**106,1299,175	≈0.40 sec.
M3RUN1	13,-145,-30	5,-145,-33	≈0 sec.	13,-145,-30	4,-110,-25	≈0 sec.
M3RUN2	13,-67,0	3,-8,-3	≈-1.00 sec.	13,-67,0	**16,-121,-63	≈0.05 sec.
M3RUN3	28,89,96	9,29,24	≈0 sec.	28,89,96	16,50,44	≈0 sec.
M3RUN4				-70,-215,6	20,63,-2	≈0.15 sec.
M3RUN5				-44,-140,99	-7,-17,16	≈0.65 sec.
M3RUN6	-11,-16,9	0,23,3	≈-0.65 sec.	-11,-16,9	3,-18,-3	≈-0.40 sec.
M3RUN7	-37,-133,10	-49,-191,19	≈0 sec.	-37,-133,-10	-5,-17,-2	≈0 sec.
M3RUN8	-22,-178,-37	-1,-7,-1	≈-0.15 sec.	-22,-178,-37	-9,-75,-16	≈-0.45 sec.
M3RUN9	-5,131,76	**465,1168,-308	≈0.55 sec.	-5,-131,-76	-20,-52,-16	≈0 sec.

** indicates the case that the predicted impact happens prior to the real one and the position vector does not have a significant meaning (see explanation in the text).

4.3 Remarks

This study shows that the trajectory can be well predicted. Estimation of the acceleration during the boosting period, the friction parameter during the non-powered period, and the navigation constant in both periods can be done from a small set of data points. Even though, with the missile type unknown, the results for 27 simulation runs show that 78% of predictions made at six seconds before the impact have the error on the time of impact smaller than 0.9 seconds. The maximal error is 2.0 seconds. The position vectors given in Table 2 are measured every 0.05 sec. They do not represent the nearest position. While the missile speed can be 2,000 to 4,500 ft/sec. In 0.05 seconds, the missile can travel 100 to 225 feet.

The execution time on a Pentium 90 MHz PC has been evaluated. In our program, one second of data were used to estimate the necessary parameters. The execution time for the task of parameter estimation is only 0.0009 seconds. The execution time for the prediction is proportional to the time period before the impact. For a 10 second period, the execution time is about 0.0093 seconds.

5. Determination of the Target among a Group of Airship

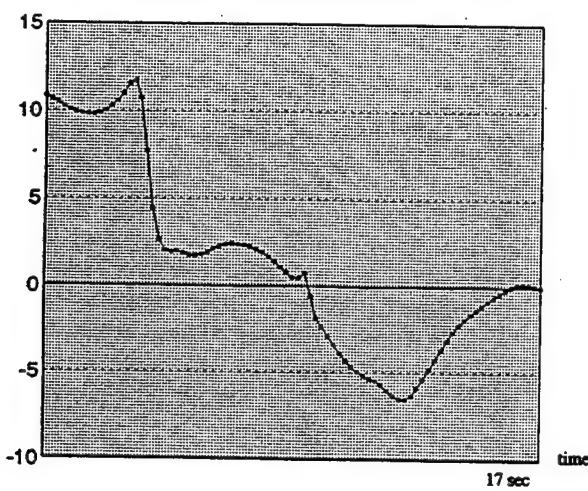
A relevant problem is to guess which airship is the most likely target of a missile if there are many airships around. Finding out the target earlier will help in creating the prompt optimum reaction. This can be done by carefully evaluating the trajectories of the potential targets and the missile. The airship that has the minimum variation of the estimated navigation constant is likely to be the target. To explain, lets consider the case M2RUN2. We added two airships beside the original one in an experiment. These two have the east coordinate offset by +1000 feet and -1000 feet, respectively. The estimated navigation constants ($\hat{\alpha}_i$) for the target and two added airships are plotted in Figure 11. The two plots in (a) and (b) for the two added airships show high irregularity and even have the value dropped below zero for a rather long period. Note that the constant should be positive. The real target has a sharp peak caused by a small ω_s within a short period. Even though

with that peak, the plot still indicates that the corresponding airplane is most likely to be the target (closest to a constant).

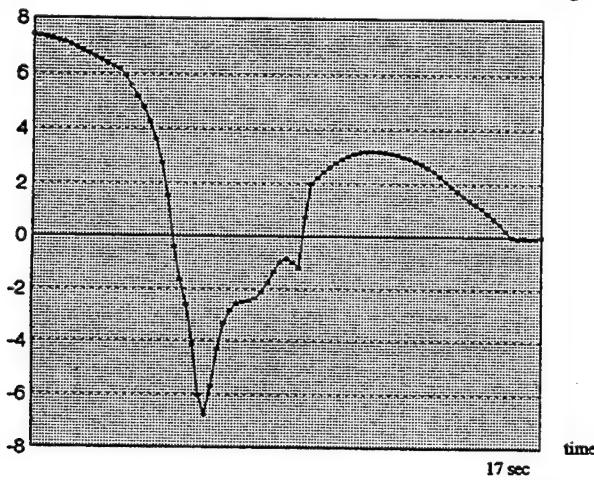
A measurement indicating if an airship is likely to be the target of an incoming missile has been defined as

$$DL_i = \begin{cases} DL_{i-1} + \min(\left| \hat{\alpha}_i - \hat{\alpha}_{i-1} \right|, 1) + \omega_s^2 & \text{if } \hat{\alpha}_i \geq 40 \text{ or } \hat{\alpha}_i \leq 0 \\ DL_{i-1} + \min(\left| \hat{\alpha}_i - \hat{\alpha}_{i-1} \right|, 1) & \text{otherwise} \end{cases}$$

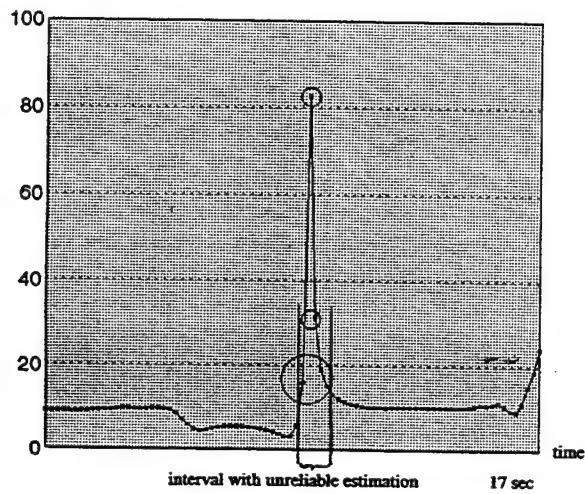
A smaller measurement of DL indicates a larger likelihood. We have tested the measurement on two selected runs M1RUN2 and M2RUN3, and plotted the results in Figure 12. Both show that the measure from the real target is the smallest. For M1RUN2, the real target is separated from another two at 12 seconds before the impact. For M2RUN3, the situation becomes clear at about 8 seconds before the impact.



(a) For the airship 1000 feet to the east of the target

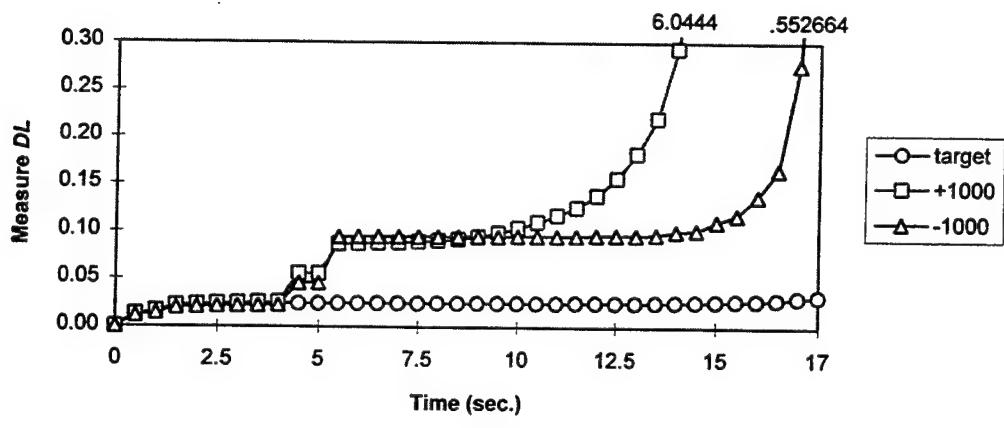


(b) For the airship 1000 feet to the west of the target

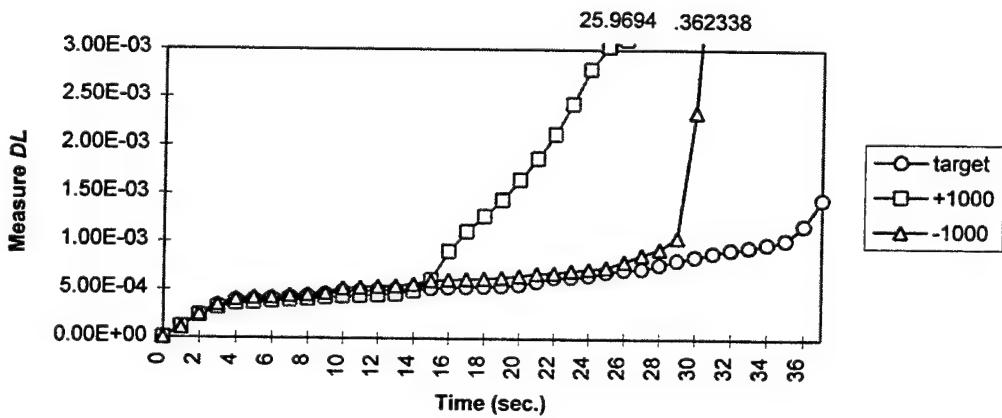


(c) For the target

Figure 11. Estimated navigation constants for target and non-targets



(a) M1RUN2



(a) M2RUN3

Figure 12. The measure DL that identifies the target

6. Conclusion

This study investigated the possibility of predicting the missile trajectory and the time of impact. The purpose is to obtain information in order to give the pilot enough time to well prepare for the best action. Since the missile trajectory is determined by the flying course of the target, the planned target path must be known to this prediction system. For a prediction made during the boosting period, the prediction technique estimates the acceleration and the navigation constant, and

uses preset values for the friction parameter and the missile peak speed for predicting the future missile trajectory. For prediction made upon burnout, the procedure estimates the friction parameter and the navigation constant for predicting the future trajectory. Tests on twenty seven sets of data from three different missiles show promising results. At six seconds before the real impact, about 78 % of predictions have the errors on the time of impact within 0.9 seconds, even though the missile type is unknown. Additional information on the missile type will generate better precision. The execution time for making one typical prediction is at the level of 10 ms on a Pentium 90 MHz PC.

References

1. Halski, D. J., R. J. Landy and R. P. Meyer, "Low Risk Evaluation and Medium Risk Development," WL-TR-94-3026, Wright Laboratory, Wright-Patterson AFB. (An ICAAS report).
2. Lin, C. S. and P. G. Raeth, "Prediction of Missile Trajectory," 1995 IEEE International Conference on Systems, Man, and Cybernetics, Vancouver, Canada, October, 1995.
3. Zarchan, Paul, *Tactical and Strategic Missile Guidance* (Volume 124 of Progress in Astronautics and Aeronautics). American Institute of Aeronautics and Astronautics, Inc. 1990.
4. Shaw, Robert L., *Fighter Combat*, Naval Institute Press, Annapolis, Maryland. 1985

Acknowledgments

The study is supported by the AFOSR 1995 SREP Subcontract 950883. Major Peter Raeth offered a lot of helps and discussions. Mr. Terry Christian and Mr. James Zeh provided the missile flyout data used in this study. Dr. Duane Warner, Mr. Gerhard Kasischke, and Mr. Chuck Plant provided detailed discussion concerning requirements for supporting active missile countermeasures. Captain Park coordinated in obtaining adequate missile flyout data.

Three-Dimensional Deformation Comparison Between F-16
Bias and Radial aircraft tires Using Optical Techniques

Paul P. Lin
Associate Professor
Department of Mechanical Engineering

Cleveland State University
Cleveland, OH 44115

Final Report for:
Summer Research Extension Program
Wright Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, Washington, D.C.

and

Cleveland State University

December 1995

THREE-DIMENSIONAL DEFORMATION COMPARISON BETWEEN
F-16 BIAS AND RADIAL AIRCRAFT TIRES USING OPTICAL TECHNIQUES

Paul P. Lin
Associate Professor
Mechanical Engineering Department
Cleveland State University

Abstract

This paper presents an optical technique called fringe projection to measure three dimensional tire deformation subjected to different loads, percentages of deflection and yaw angles. Unlike the well-known Moire method, the proposed technique uses a single light source and one grating, thus requiring no image superposition. As a result, the measurement is not as sensitive to vibration as the Moire method. The fringe projection also differs from the commonly used optical inspection technique in manufacturing industry via line scanning known as structured light, which cannot be applied to dynamic deformation measurements. The recently developed subpixel resolution was employed to accurately locate the optical fringe centers, which in turn improves the accuracy in 3-D geometry determination. A fiber-optic displacement sensor was also placed close to the tire sidewall in order to measure the deformational change of the selected reference point. Finally, the deformations are compared between F-16 bias and radial aircraft tires when subjected to the same loading conditions.

THREE-DIMENSIONAL DEFORMATION COMPARISON BETWEEN
F-16 BIAS AND RADIAL AIRCRAFT TIRES USING OPTICAL TECHNIQUES

Paul P. Lin

Introduction

In non-contact measurement, several optical techniques are available. Laser ranging can yield a dense set of depth values with which surface structure can be obtained through surface fitting or approximation (Vemuri and Aggarwal, 1984). This technique, however, is usually slow and expensive. Stereo vision utilizes the disparity between the projected positions of a point in two images to infer the depth of this point (Marr and Poggio, 1976). But the correspondence between points in the stereo images is difficult to establish, and the computation is sensitive to errors introduced in digitization and camera calibration. The well known Projection Moire technique uses a white light or laser light source and two gratings of the same pitch (one in front of light and the other one in front of camera) to generate Moire interference patterns. The image is recorded in a single CCD camera, in lieu of two cameras used in stereo vision. Another very similar technique, Shadow Moire, uses only one grating near the object to generate the Moire patterns. The Moire contours thus obtained, however, do not make a difference between peaks and pits unless prior information or additional algorithms are applied. Furthermore, this technique is very sensitive to vibration due to the necessity of superimposing two images. The most accurate optical technique available today is phase-shifting interferometry (PSI). It takes time to generate three or four consecutive phase shifts to form interferograms, and thereby rendering PSI not useful for measurement of dynamic motion. This technique, by nature is also very sensitive to vibration. Another consideration is that Moire technique requires the use of two very fine pitch gratings (usually over 250 lines per inch), which makes it very difficult to visualize the generated Moire pattern on a low reflectivity aircraft tire.

The proposed fringe projection technique (Lin & Parvin, 1990; Lin, et. al., 1991, Lin & Kuo, 1995) uses a single light source and a grating of 100 lines per inch in front of light projector to generate many optical fringes which cover the object under test. No image superposition is required. In comparison with the Moire or phase-shifting technique, the fringe projection technique is less sensitive to vibration and much more computationally efficient. Although the well-known structured light and the fringe projection share the same concept of triangulation, they differ from each other in the way of determining the 3-D geometry. The latter captures an image containing all fringes in nearly real-time (1/30 seconds) and utilizes the spacing information between fringes, whereas the former projects fringes one at a time and move the object at a constant speed in order to scan the necessary portion of the object. The structured light cannot be used for tire deformation statically or dynamically. Under the static loading situation, first, the tire pressure and applied load cannot be physically held constant as a function of time, second, the tire sidewall is too big to scan.

Optical measurement systems usually assume a fixed reference point on which three dimensional geometry is based on. In contrast, tire deformation measurement is rather challenging, not only the tire rotates and translates in order to reach the preset percentage of deflection but also deforms. Thus, it is necessary to install a device that moves with the tire and continuously monitors the surface height change of the selected reference point. This paper first describes the optical measuring system, then the accurate fringe detection technique. In the section of results and discussion, the 3-D tire deformation data and the comparison between these two types of tires will be discussed. At the end, some conclusions will be drawn.

The Measuring System

The measuring system consists of

- (1) Optical equipment: White light projector, grating,

optical rails, CCD camera and close-range fiber-optic displacement sensor.

- (2) Image acquisition equipment: Frame grabber, image acquisition and processing software.
- (3) Data acquisition equipment: Dual-channel digital data storage oscilloscope.
- (4) High Speed Flash: Microseconds flash duration with a few nanoseconds response time.
- (5) Synchronization Device: Synchronize the tire rotation with the flash.
- (6) Recording equipment: Super VHS video recorder (VCR).
- (7) Computing equipment: 486-based micro-computer and RGB monitor.

The light produced by the white light projector passes through the grating (Ronchi ruling) and illuminate the tire surface (see Figure 1). The image captured by the CCD camera is recorded in the VCR, and then transmitted to the frame grabber where the image data are digitized and processed. The digital image is then displayed in a high resolution RGB monitor. The frame grabber and the CCD camera both have the same resolution of 512 by 480 pixels. The highest shutter speed available in this camera is 100 micro seconds. The camera, frame grabber and VCR's frame rate is 1/30 seconds. The pitch of the projection grating used was 100 lines per inch and it was placed close to the tire so as to produce about 11 optical fringes when the tire was unloaded. Furthermore, the use of a new CCD camera with 2/3 inch imager made it possible to be 1/3 closer to the tire when comparing with the most popular size of imager: (1/2 inch). This arrangement is particularly important for a low reflectivity object such as tires.

It is necessary to specify the location of the reference point within the tire and near the wheel flange. In addition, a reference plane (xy plane) passing through this point and perpendicular to the viewing direction has to be established. It is worthwhile to note that when a tire is loaded, not only the location (x and y components) of the reference point changes, the surface height (z component) of the point (i.e. perpendicular to the sidewall) changes as well.

In this study, a close-range fiber-optic sensor was installed to accurately measure the reference point's height change when a tire is loaded. During the tire loading process, this sensor moved with the tire in order to keep the detecting position constant. When performing image analysis, fringe 1 is scanned first from top to bottom and the remaining fringes are then scanned horizontally from left to right and top to bottom (see Figure 2).

A wobbling motion usually occurs when a tire rotates against a flywheel. However, it was found that the degree of wobble was very small at the Air Force tire test facility (dynamometer 120). Thus, there was no need to synchronize the tire rotation with a flash. Instead, a projector producing continuous light and a CCD camera set at a shutter speed of 1/1000 seconds were used for better image quality. A synchronization device was prepared just in case if wobbling ever occurs. It was designed to turn on the high speed flash exactly once per revolution so that same points of interest in the tire can be captured by the CCD camera. A reflective tape is placed on the reference point and the probe of the fiber-optic sensor is then focused on the tape. Initially, the height of this point is detectable. As the tire rotates, the tape goes away from the fiber-optic light beam which results in an out-of-range detection with zero output voltage. As the tire comes back to the same angular position (i.e. exactly one revolution), the voltage will be back to the initial value. Since the voltage generated by the fiber-optic sensor changes so fast that it is necessary to use a digital data storage oscilloscope to store the digital data for later examination. A threshold voltage near the peak value can be set so as to trigger the flash. Thus, the flash is fired only once per revolution. The flash takes a few nanoseconds of response time and 9 microseconds of flash duration operating at the power of 60 Watts. During our tests, the synchronization device actually worked very well. The only concern was that the light energy reduces when the flashing frequency increases. In many cases, the acquired images appeared to be a little too dark to be used for image analysis.

The acquired images were filtered and analyzed. To determine the three

dimensional geometry of tire deformation, the in-house developed computational algorithms were employed (Lin and Kuo, 1995). With non-zero yaw angles, the computational algorithms were modified to consider the effect of imaging distance on dimension determination. More specifically, with the optical arrangement as shown in Fig. 3, half the loaded tire is closer to the camera while the other half is farther, instead. With yaw angles, two referenced planes both passing through the tire horizontal center line are established: the primary plane which is perpendicular to the camera's viewing direction and the secondary plane which is inclined with a yaw angle from the primary plane. The tire deformation to be determined is based on the secondary plane which is the tire sidewall with a yaw angle.

Accurate Fringe Center Detection via Subpixel Resolution

The principle of the 3-D optical measurement used here is based on the curvature change of projected fringes and the spacing between two adjacent fringes. Thus, the first step in image analysis is to accurately detect the locations of fringe centers, line by line. The accuracy of three dimensional geometry determination greatly depends upon the accuracy of fringe center detection. Usually the fringe center locations are limited to one-pixel resolution which is provided by the CCD camera. In this research, the so-called sub-pixel resolution was employed in order to improve the fringe center detection accuracy (Lin and Parvin, 1990).

At first, a low-pass filter is used to remove noisy pixels and then an optimum reconstruction filter is chosen to prevent aliasing effect and reduce the amount of necessary computation. With the grating standing vertically, the light intensity distribution horizontally across a fringe or stripe was closely examined. It was found that the distribution very much resembles a Gaussian distribution, and the exact fringe center should be located at the maximum of the distribution curve. The optimum reconstruction filter for a Gaussian distribution is chosen to correspond to a standard deviation of 1.0 pixel period.

In the spatial domain, the convolution sum for the reconstruction consists of only five terms. The probability density function for the Gaussian distribution is given by

$$g(x) = [\exp(-0.5x^2)] / \sqrt{2\pi} \quad (1)$$

where *exp* stands for exponential function with the mean value of zero and the standard deviation of 1.0.

Since only five terms are necessary for computation, the fringe center can be approximately located using one-pixel resolution as the central position (i.e. position 0). Two positions prior to and after position 0 are numbered -2, -1, 1 and 2, respectively. Thus, a function $f(x)$ can be determined by taking five terms in the convolution sum into account.

$$f(x) = \sum_{j=-2}^{2} f_j \exp[-0.5(x-j)^2] \quad (2)$$

where f_j is the pixel value at position j , and x is the position to be used for computation. As mentioned earlier, the exact fringe center is located at the maximum value of a Gaussian distribution curve. Taking the first derivative of the $f(x)$ and using the bisection numerical method to find the root of $f'(x)=0$ was found to be the most reliable method for finding the maximum value within a specified interval.

Results and Discussion

In this research, F-16 bias and radial tires were tested under the same loading conditions statically and dynamically. Both tires were loaded against a flywheel with 30% deflection at 0° , 2° , $\pm 4^\circ$ yaw angles. The tire rotating speeds are 10 mph and 40 mph at the various yaw angles. Furthermore, the tires were also tested under the conditions of maximum-power takeoff, landing-taxi and taxi-refused takeoff at 0° yaw angle. Table I shows that both tires require different loads to produce the same deflection. In the tests of maximum-power takeoff, landing-taxi and taxi-refused takeoff, both tires had to follow the

same load and speed profiles as initially programmed. In order to have a meaningful comparison between two tires, the initial pressure on the radial tire was raised from 310 psi to 390 psi.

TABLE I

Loading Comparison between F-16 Bias and Radial Tires

Loading Condition: Rated Pressure (310 psi initially)
Corrected Load (as shown below)

<Subjected to Flywheel Loading only>

(A) Static, 10 and 40 MPH

Deflection	Bias	Radial
30 %	14000 lb (initially 310 psi)	12000 lb (initially 310 psi)

(B) Maximum-Power Takeoff

Deflection	Bias	Radial
33 %	16200 lb (initially 310 psi)	16200 lb (initially 390 psi)

It can be seen from Table I that the radial tire requires about 16% less load than the bias tire for producing 30% deflection. In other words, when subjected to the same load, it is easier to deflect a radial tire. The 3-D deformation under study is a small but significant portion of the deformed tire (a rectangular region near the contact between tire and flywheel). Fig. 4 indicates this portion with the shaded area as well as the locations of two reference points. The first selected reference point was located on the tire sidewall, near and above the wheel flange and approximately at 11 O'clock position (far away from the loading contact area), at which the height change or the out-of-plane deformation was measured directly by the fiber-optic displacement sensor. This height change is perpendicular to the tire sidewall in the direction parallel to the wheel axle.

Figs. 5, 6, 7 and 8 show some of the 3-D deformation data for these two tires. Tables II and III show the summarized deformations subjected to various

loading conditions. Each deformation shows the magnitude between the minimum and maximum (min-max) across a row of interest, relative to the second reference point (at the upper-left corner of the shaded area). Afterward, the absolute deformation can be computed by adding the additional deformation of the second reference point itself to all the min-max deformation data.

As can be seen from Tables II and III, when subjected to the same percentage of deflection, the deformation magnitudes appear to be about the same between these two tires, but the shape of deformation is different. Unlike the bias tire, the radial tire looks like a flat tire when loaded. The conclusions based on the deformation data analysis are summarized in the next section.

Conclusions and Future Work

The measuring system and the optical technique used worked very well. The selected camera shutter speed of 1/1000 seconds was adequate. Various yaw angles and tire rotation speeds were implemented during the dynamic tests, but their complete deformation analysis results are not yet available. In this paper, only the tire deformations under the static loading conditions with and without yaw angles are included.

After analyzing all deformation data near the contact area, the following conclusions can be made:

- (1) In general, both tires exhibit approximately the same min-max deformation magnitudes, even though the shapes of deformation are different, although the radial tire seems to deform a little more.
- (2) For both tires, the degree of deformation increases as the percentage of deflection increases (or as the applied load increases).
- (3) The presence of yaw angle has some effect on both tires in terms of deformation magnitude. The radial tire deforms a little more than the bias tire

with yaw angles, but more symmetric with respect to positive and negative yaw angles. In contrast, the bias tire deforms a little more with positive yaw angle than negative one. Furthermore, the bias tire shows a proportional relationship between the relative min-max deformation magnitude and the yaw angle, the magnitude increases as the yaw angle in the positive direction increases. In contrast, the radial tire exhibits relatively insensitive to yaw angle change.

(4) In terms of maximum absolute deformation, bias tires exhibit little more than the radial ones, but the difference is insignificant. Note that this is not the relative min-max deformation as shown in Tables II and III. Rather, it is the maximum overall deformation when compared to the free loading condition.

Future work will include 3-D deformation analysis of dynamic testing. It will be interesting to see how rotating speed affects tire deformation, especially at critical speeds.

References

Lin, P. P. and Parvin, F., (1990) "Edge Detection with Subpixel Resolution and its Application to Radius Measurement via Fringe Projection Technique," SME Technical Paper, MS90-576, pp. 4-13 - 4-27.

Lin, P. P., Parvin, F., and Schoenig, Jr., F. C., (1991) "Optical Gaging of Very Short-term Surface Waviness," Transactions of NAMAR/SME, pp. 327-322.

Lin, P. P., Chawla, M. D., and Ulrich, P. C., (1994) "Optical Technique for Measuring Tire Deformation and Strains - Preliminary Results," SAE Technical Paper No. 941178, Aerospace Atlantic Conference and Exposition.

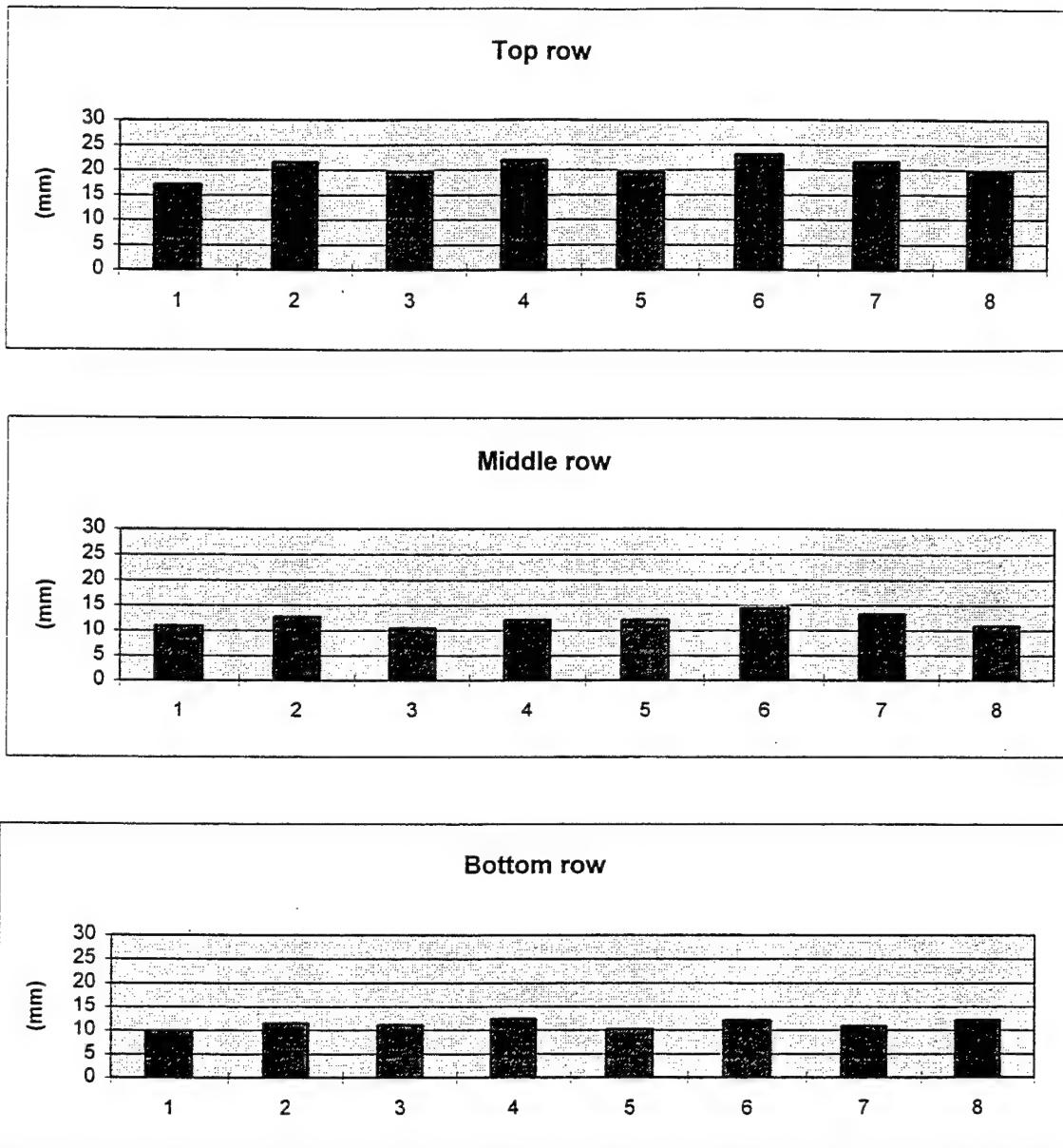
Lin, P. P., Chawla, M. D., and Wagner, P. M., (1995) "Deformation Comparison between Bias and Radial Aircraft Tires Using Optical Techniques", SAE Technical Paper No. 951433, Aerospace Atlantic Conference and Exposition.

Lin, P. P. and C. T. Kuo (1995) "Efficient Surface Geometry Measurement by Means of Optical Fringe Projection," Manufacturing Science and Engineering, MED-Vol. 2-1/MH-Vol. 3-1, American Society of Mechanical Engineering, pp. 507-518.

Marr, D. and Poggio, T., (1976) "Cooperative Computation of Stereo Disparity," Science, V. 194, pp. 283-287.

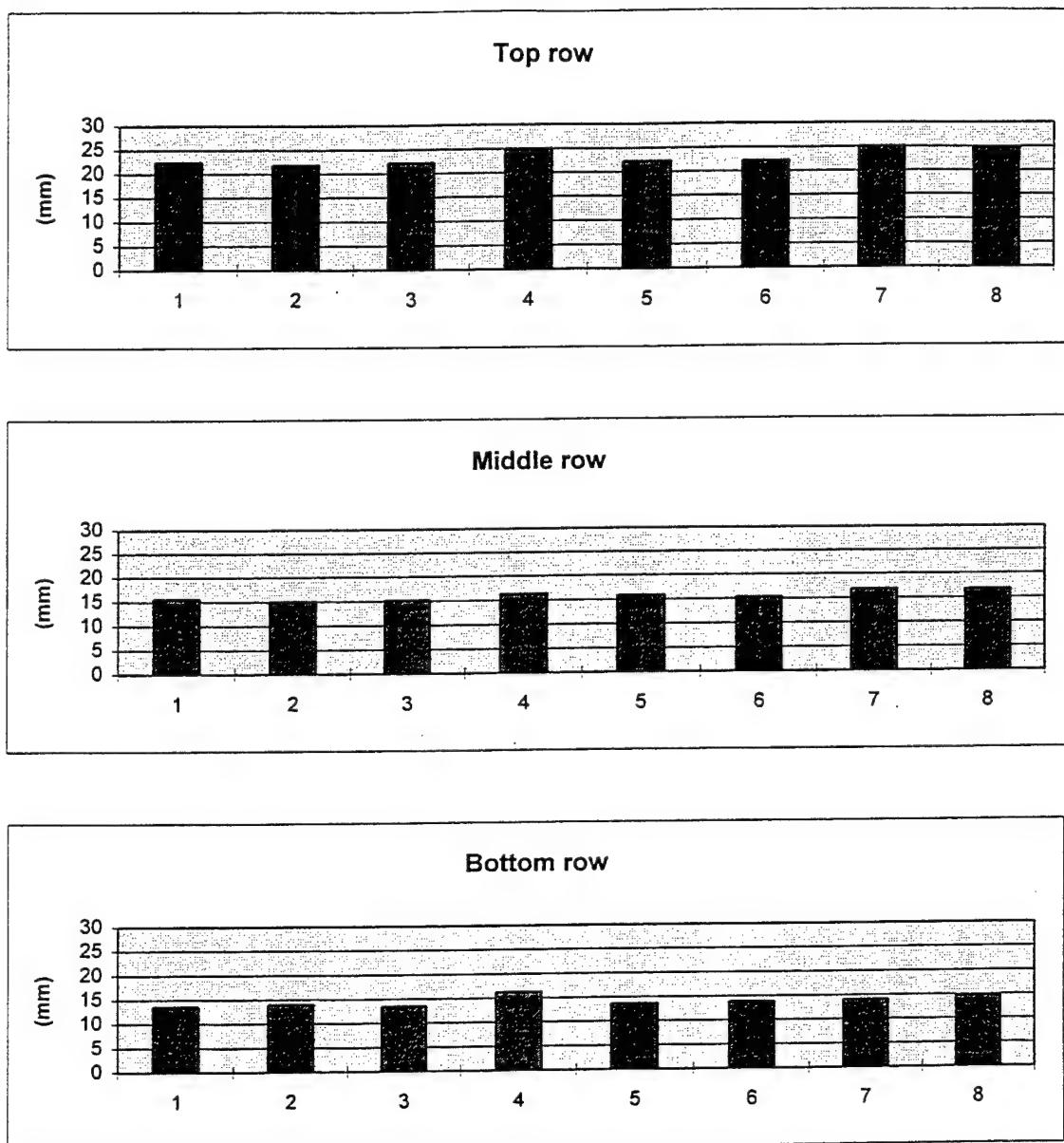
Vemuri, B. C. and Aggarwal, J. K., (1984) "3-Dimensional Reconstruction of Objects from Range Data," Proc. of 7th Int. Conf. on Pattern Recognition, V1, pp. 752-755.

TABLE II
Bias Tire: Relative Deformation (Min-Max)



1	0% deflection, 0 mph, yaw=0°	5	0% deflection, 0 mph, yaw=4°
2	30% deflection, 0 mph, yaw=0°	6	30% deflection, 0 mph, yaw=4°
3	0% deflection, 0 mph, yaw=2°	7	0% deflection, 0 mph, yaw=-4°
4	30% deflection, 0 mph, yaw=2°	8	30% deflection, 0 mph, yaw=-4°

TABLE III
Radial Tire: Relative Deformation (Min-Max)



1	0% deflection, 0 mph, yaw=0°	5	0% deflection, 0 mph, yaw=4°
2	30% deflection, 0 mph, yaw=0°	6	30% deflection, 0 mph, yaw=4°
3	0% deflection, 0 mph, yaw=2°	7	0% deflection, 0 mph, yaw=-4°
4	30% deflection, 0 mph, yaw=2°	8	30% deflection, 0 mph, yaw=-4°

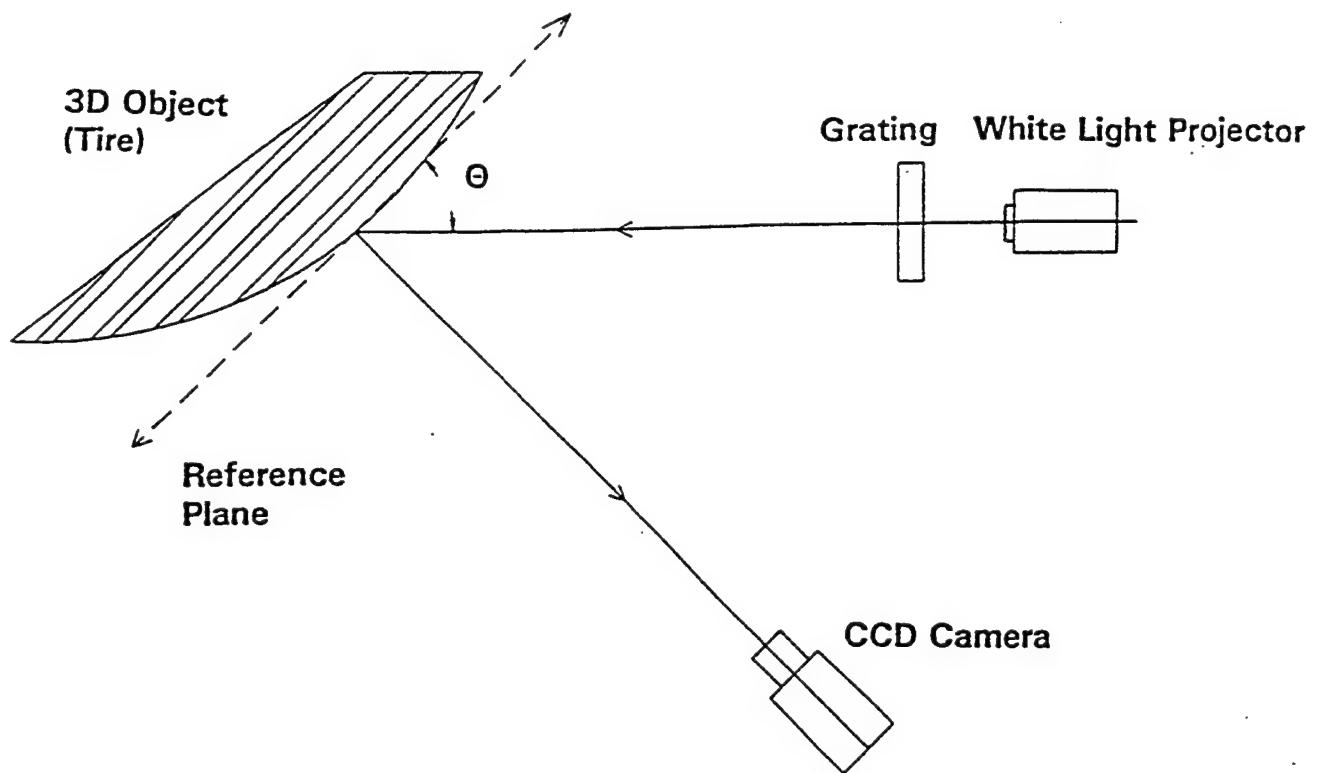
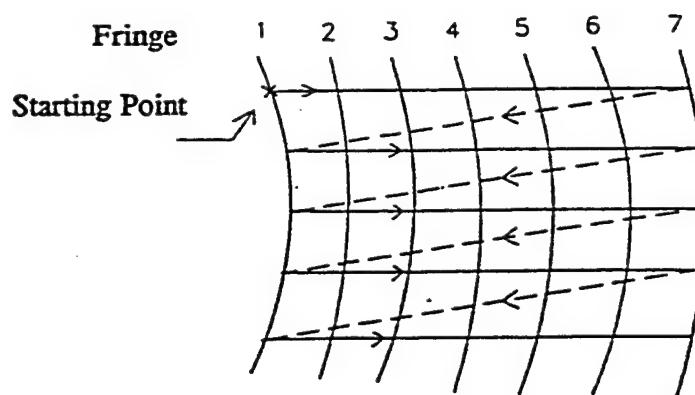


Fig.1 Experimental Arrangement for 3D Geometry Measurement



Note: Fringe 1 is scanned first (from top to bottom) and the remaining fringes are then scanned horizontally line by line as shown in this figure.

Fig. 2 Image Data Scanning Process

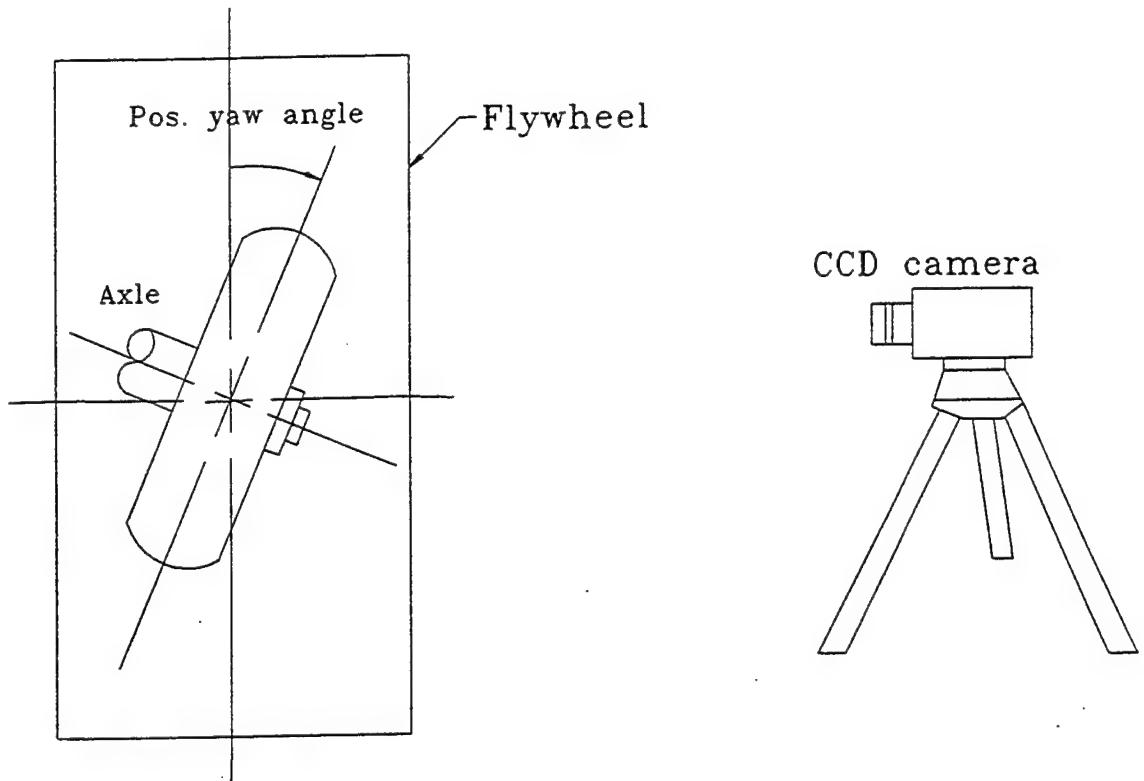


Fig. 3 Loaded Tire with Yaw Angle

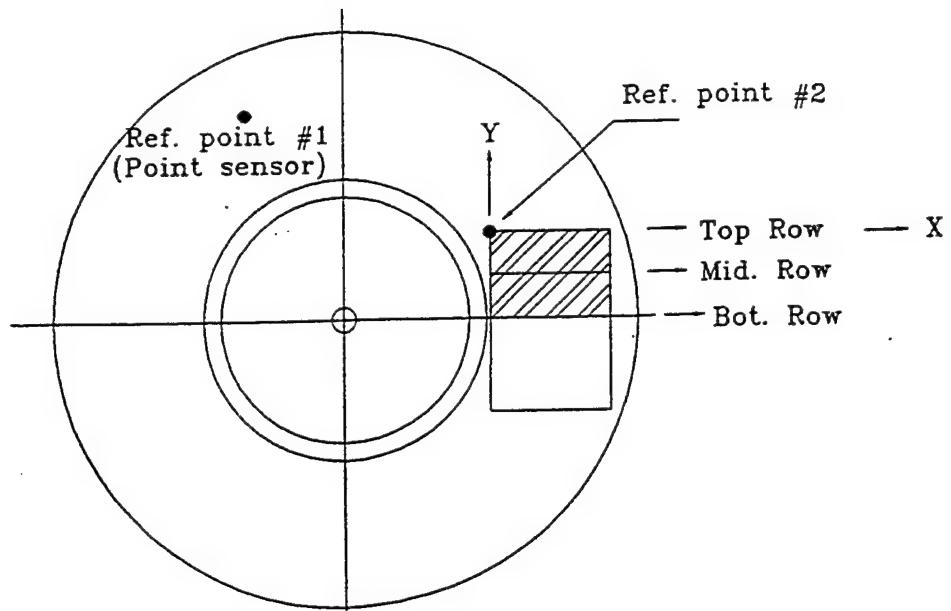
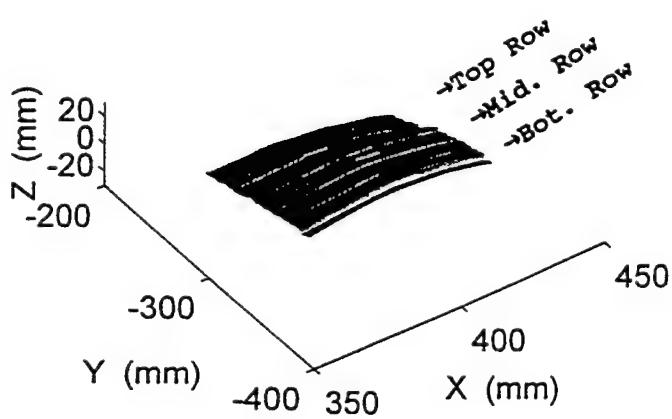
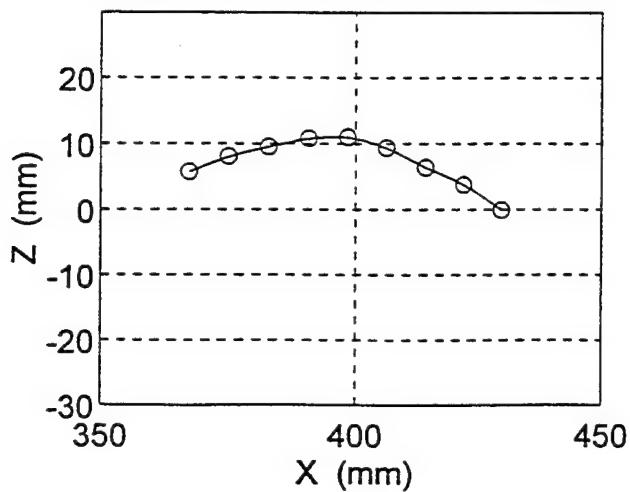


Fig. 4 Reference Points and the Area of Interest

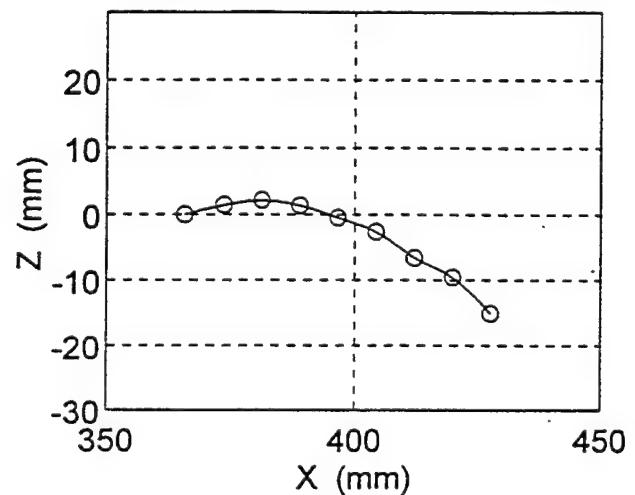
F16 Bias: 0% Deflection, 0 mph
Initial Tire Pressure: 310 psi



Deformation at Middle Row



Deformation at Top Row



Deformation at Bottom Row

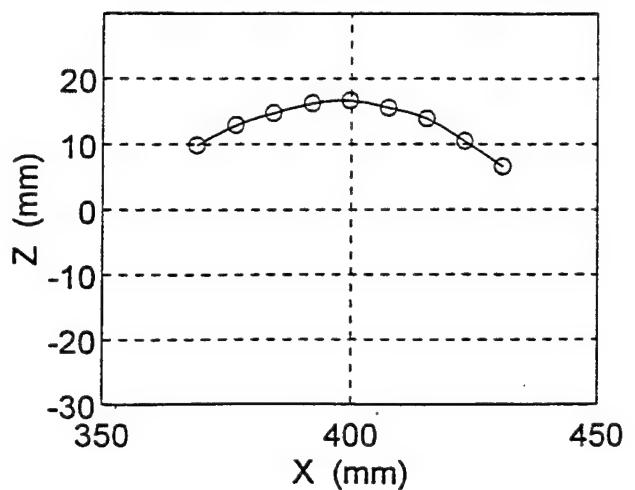
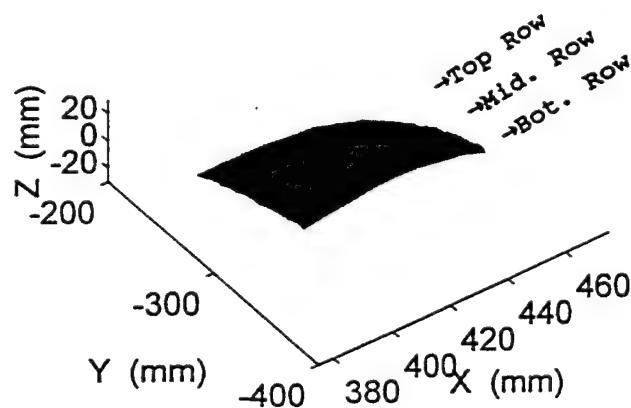
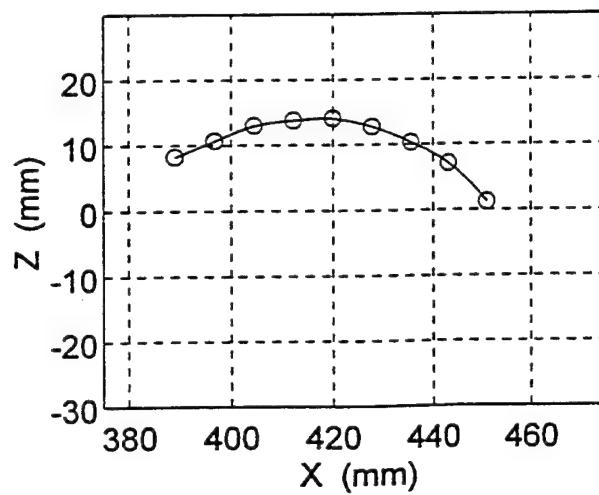


Fig. 5 F-16 Bias Tire with 0% Deflection and 0° yaw

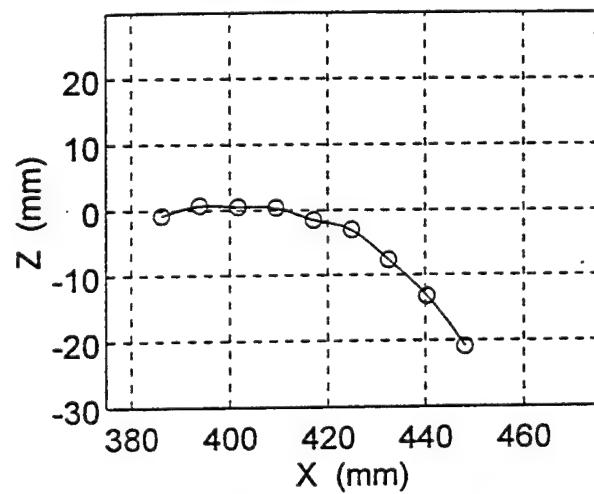
F16 Bias: 30% Defl., 0 mph
Load: 12000 lb, Pressure: 317 psi



Deformation at Middle Row



Deformation at Top Row



Deformation at Bottom Row

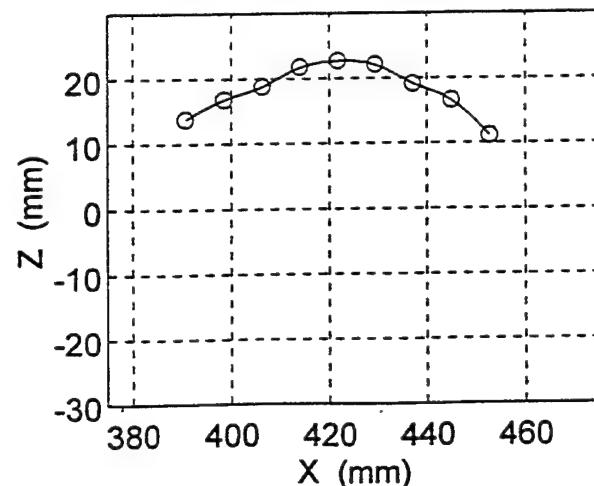
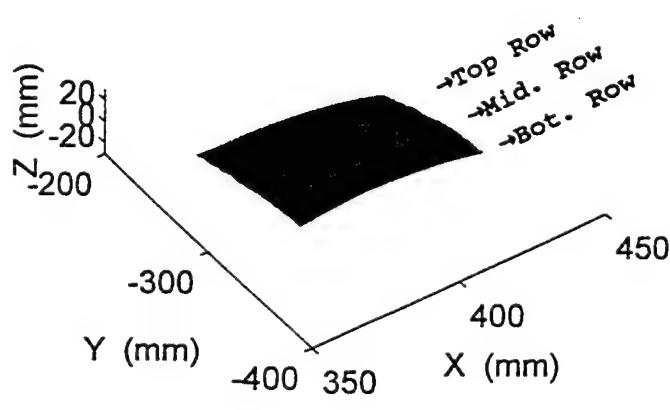
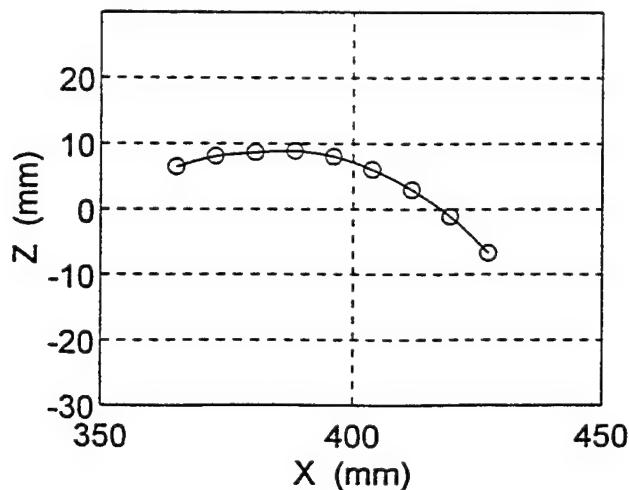


Fig. 6 F-16 Bias Tire with 30% Deflection and 0° yaw

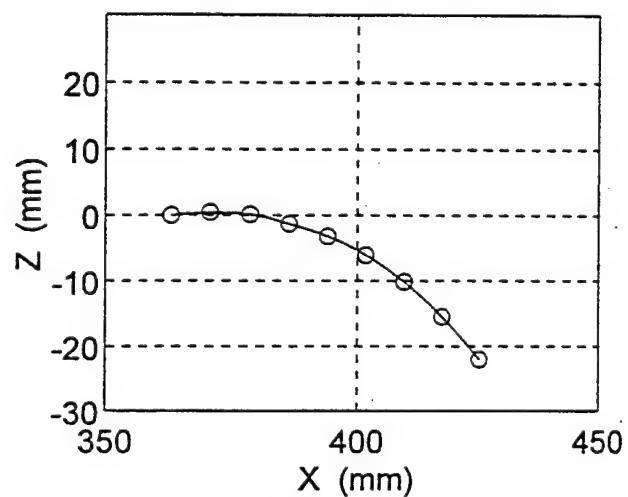
F16 Radial: 0% Deflection, 0 mph
Initial Tire Pressure: 310 psi



Deformation at Middle Row



Deformation at Top Row



Deformation at Bottom Row

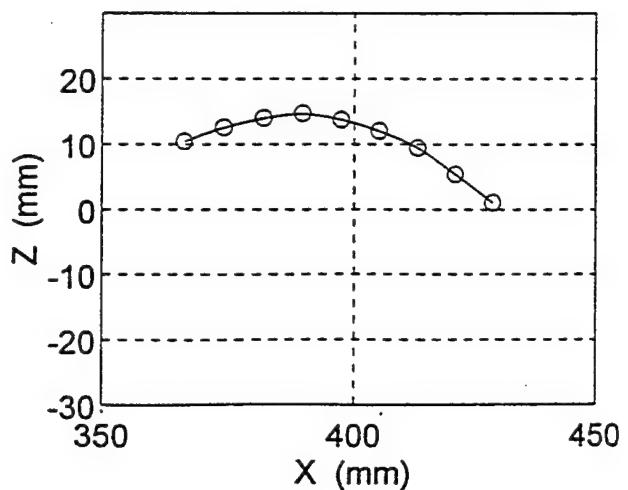
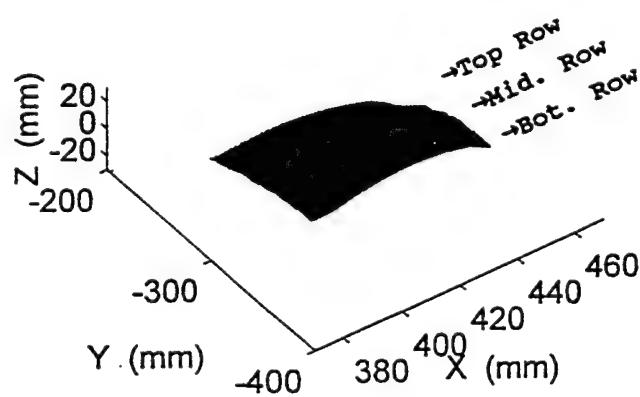
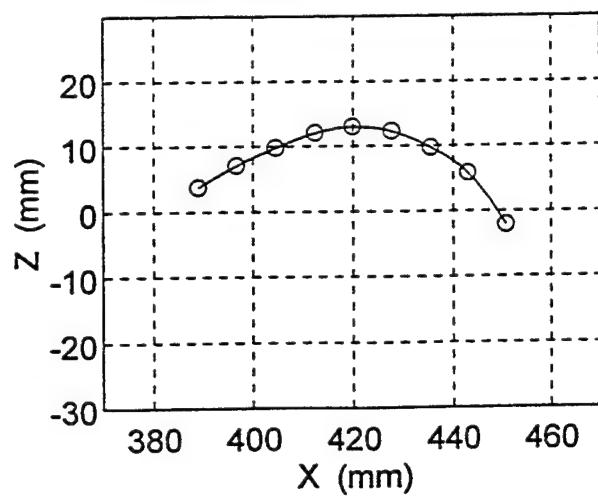


Fig. 7 F-16 Radial Tire with 0% Deflection and 0° yaw

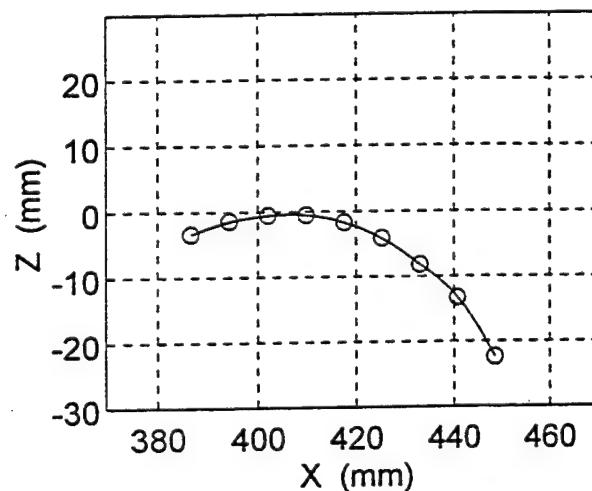
F16 Radial: 30% Defl., 0 mph
Load: 12000 lb, Pressure: 317 psi



Deformation at Middle Row



Deformation at Top Row



Deformation at Bottom Row

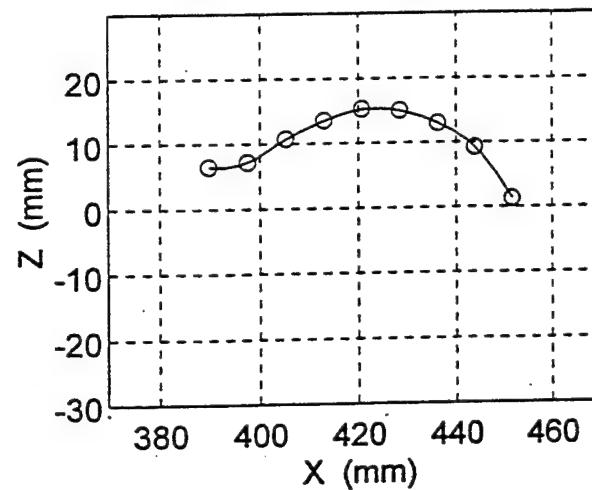


Fig. 8 F-16 Radial Tire with 30% Deflection and 0° yaw

**MODELING THE POST-BURN-IN ABNORMAL BASE CURRENT IN
AlGaAs/GaAs HETEROJUNCTION BIPOLAR TRANSISTORS**

**J. J. Liou
Associate Professor and Graduate Coordinator
Department of Electrical and Computer Engineering**

**University of Central Florida
Orlando, Florida 32816**

**Final Report for:
Summer Faculty Research Program
Wright Laboratory**

**Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC**

and

Wright Laboratory

December 1995

MODELING THE POST-BURN-IN ABNORMAL BASE CURRENT IN
AlGaAs/GaAs HETEROJUNCTION BIPOLAR TRANSISTORS

J. J. Liou

Associate Professor and Graduate Coordinator
Department of Electrical and Computer Engineering

Abstract

The base current of AlGaAs/GaAs heterojunction bipolar transistor subjected to a long burn-in test often exhibits an abnormal characteristic with an ideality factor of about 3, rather than a normal ideality factor between 1 to 2, in the mid-voltage range. This paper develops an analytical model to investigate the physical mechanisms underlying such a characteristic. Consistent with the finding of an experimental work reported recently, our model calculations show that the recombination current in the base has an ideality factor of about 3 in the mid-voltage range and that such a current is responsible for the observed abnormal base current in heterojunction bipolar transistor after a long burn-in test. Post-burn-in data measured from two different heterojunction bipolar transistors are also included in support of the model.

MODELING THE POST-BURN-IN ABNORMAL BASE CURRENT IN
AlGaAs/GaAs HETEROJUNCTION BIPOLAR TRANSISTORS

J. J. Liou

1. INTRODUCTION

Burn-in tests carried out in a thermal and/or electrical stress condition are useful in determining the long-term performance of AlGaAs/GaAs heterojunction bipolar transistors (HBTs) [1-3]. Experimental results often show that the burn-in test increases considerably the base current I_B but does not alter notably the collector current I_C . Furthermore, an abnormal base current with an ideality factor $n \approx 3$ in the mid-voltage range is often observed in the Gummel plot of an HBT subjected to a relatively long-hour burn-in test [1-3]. An attempt has been made earlier to model the HBT post-burn-in behavior [4]. The analysis was based on the theory that the defects at the base surface may migrate to the hetero-interface during the high thermal/electrical stress condition (i.e., recombination/thermal enhanced defect diffusion [5]). While such a model can successfully describe I_B and I_C in HBTs subjected to a relative short burn-in test (144-hrs stress in Fig. 1), it fails to predict I_B with $n \approx 3$ characteristics observed in the HBT after a long-hour stress test, as evidenced by the results of I_B measured after 300-hrs stress. Sugahara et al. [3] have suggested that such an abnormal current can be attributed to a significant increase in the number of defects in the strained base (i.e., stress-induced defects) during the long stress hours. Also, they have demonstrated that the post-burn-in I_B can be greatly reduced if the base lattice strain is relaxed.

This paper presents a comprehensive theoretical study on the abnormal base current in the post-burn-in HBT. Based on the Shockley-Read-Hall (SRH) recombination statistics, a model for the recombination current in the base region is developed. Our model calculations show that such a current has an ideality factor of about 3 in the mid-voltage range and thus is responsible for the observed abnormal base current in HBT after a long burn-in test. With the aid of the model and measurement data, physical mechanisms underlying the observed abnormal base current in the post-burn-in HBT are also discussed.

2. MODEL DEVELOPMENT

2.1 Pre-Burn-In HBT

We focus on the base current of a mesa-etched N/p⁺/n HBT. There are two major components for the base current of pre-burn-in HBT:

$$I_B = I_{BL} + I_{BN} \quad (1)$$

where I_{BL} is the base leakage current and I_{BN} is the normal base current. For the bias condition of applied base-collector voltage $V_{CB} = 0$ and base-emitter voltage $V_{BE} > 0$ (i.e., forward-active mode), the base leakage current is originated from the leakage of electron from the base to emitter through the emitter-base periphery and is the

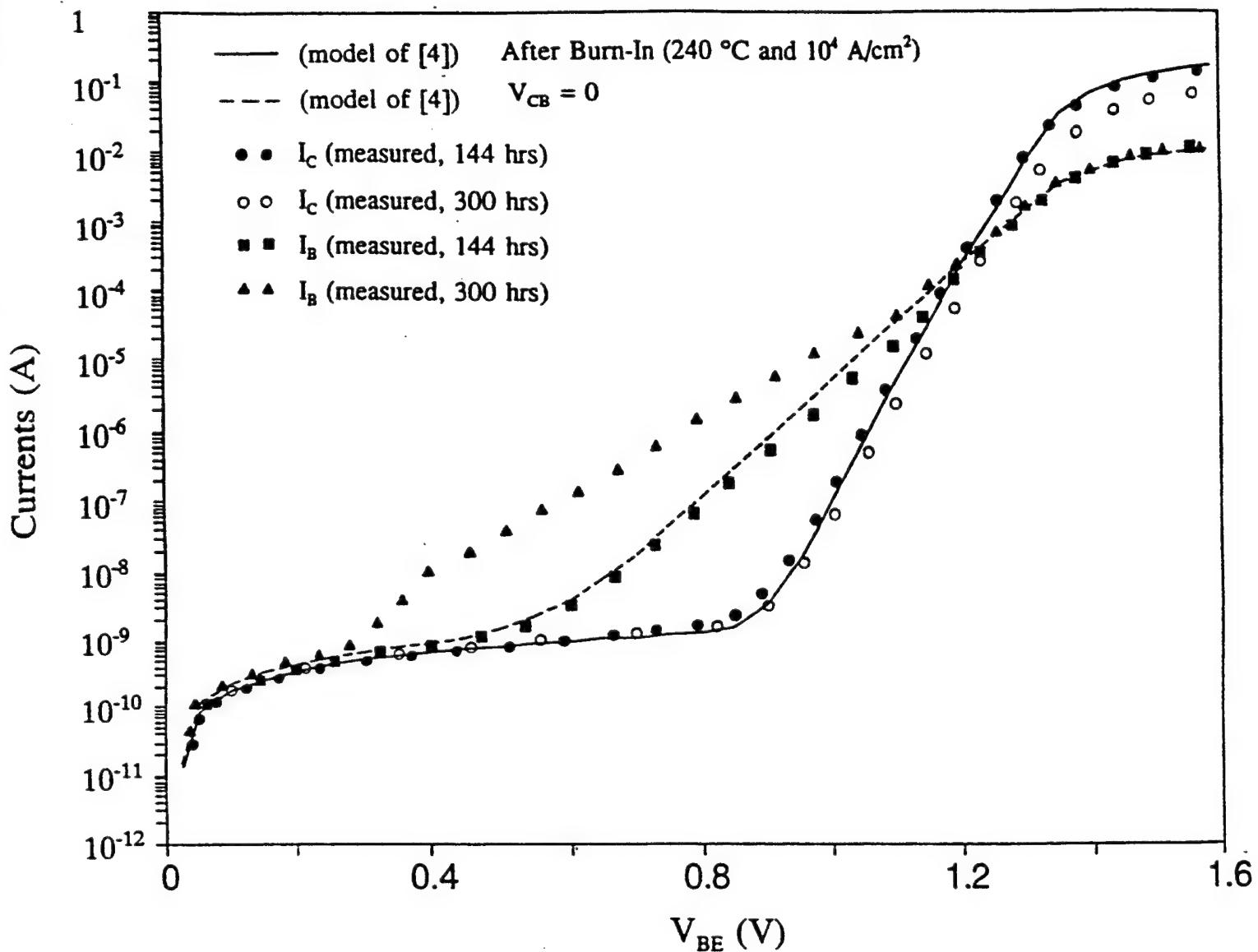


Fig. 1 Base and collector current characteristics of a post-burn-in (subjected to 240 °C temperature and 10^4 A/cm 2 current density stress) AlGaAs/GaAs HBT calculated from a previously developed model [4] and obtained from measurements. The model of [4] gives an accurate prediction for the HBT behavior after a relatively short stress test (i.e., 144 hrs), but fails to describe the base current (i.e., with an ideality factor of about 3 between $V_{BE} = 0.3$ and 1.2 V) of the HBT subjected to a long stress test (i.e., 300 hrs).

dominate current component for I_B at relatively small V_{BE} [6]. This current is given by [6]

$$I_{BL} = P_E J'_{BL} [1 - \exp(-V_{BE} F_L / V_T)] \quad (2)$$

where P_E is the emitter perimeter length, J'_{BL} is the fully-activated (i.e., $V_{BE} \gg V_T$) base leakage current density, and F_L is an empirical parameter determining the shape of the base leakage current.

The normal base current in general consists of 1) the recombination current I_{SCRE} in the emitter-side of the heterojunction space-charge region; 2) the recombination current I_{SCRB} in the base-side of the heterojunction space-charge region; 3) the surface recombination current I_{RS} at the emitter side-walls and extrinsic base surface; 4) the recombination current I_{QNB} in the QNB; and 5) the injection current I_{RE} from the base into emitter. The details of these current components can be found in the literature [7]. For pre-burn-in HBTs, I_{QNB} is negligible because the number of defects in the QNB is small and the base is very thin. In addition, I_{SCRB} is neglected due to the fact that the majority of space-charge region (SCR) resides in the emitter because of the very high base doping density. Thus

$$I_{BN} = I_{SCRE} + I_{RS} + I_{RE} \quad (3)$$

The ideality factor of this current ranging from 1 to 2.

2.2 Post-Burn-In HBT

After a long burn-in test, the number of defects in the base will be increased significantly due to the strained lattice during the stress test [3]. As a result, substantial electron-hole recombination occurs in both the base-side of the SCR and the QNB, and the conventional thin- QNB and thin-SCR approximations are no longer valid. Thus, for an HBT after a long burn-in test,

$$I_{BN} = I_{BASE} + I_{SCRE} + I_{RS} + I_{RE} \quad (4)$$

where $I_{BASE} = I_{SCRB} + I_{QNB}$, and

$$I_{BASE} = Aq \int_0^{X_2} U^{SRH}(x) dx + Aq \int_{X_2}^{X_B} U^{SRH}(x) dx \quad (5)$$

Here A is the emitter area, $x = 0$ and X_2 are the boundaries of base-side SCR, $x = X_2$ and X_B are the boundaries of the QNB, and U^{SRH} is the total SRH recombination rate summing the recombination rates at each trapping state

E_{Ti} ($i = 1, 2, \dots, N$, N is the total number of trapping states):

$$U_{SRH}^{\text{SRH}} = \sum_{i=1}^N U_i^{\text{SRH}} \quad (6)$$

and [7]

$$U_i^{\text{SRH}} = (pn - n_i^2)(1 + \Gamma)(N_{Ti}\sigma_i v_{th})/\{p + n + 2n_i \cosh[(E_{Ti} - E_i)/kT]\} \quad (7)$$

p and n are hole and electron concentrations in the QNB, n_i is the intrinsic free-carrier concentration, Γ is the trap-assisted tunneling factor, N_{Ti} is the trapping density at E_{Ti} , σ_i ($\sim 10^{-14} \text{ cm}^2$) is the capture cross section at N_{Ti} , v_{th} ($\sim 10^7 \text{ cm/sec}$) is the electron thermal velocity, and E_i is the intrinsic Fermi energy. The trap-assisted tunneling is important for high field region, such as the emitter-base SCR, where electrons can tunnel through the energy band via traps and subsequently recombine with holes [8]. In a low field region, such as the QNB, Γ approaches zero. This factor is given by [8]

$$\Gamma = (\Delta E/kT) \int_0^1 \exp(u\Delta E/kT - K'u^{1.5}) du \quad (8)$$

Here ΔE is the energy between the conduction band edge and the trapping state energy since electrons in these energies are tunneling possible, and K' is a parameter inversely proportional to the local electric field ξ :

$$K' = (4/3)(2m^* \Delta E^3)^{0.5}/(q\hbar\xi) \quad (9)$$

m^* is the effective electron mass and \hbar is the reduced Planck constant. When ξ is large, K' is small, and Γ becomes large.

For the QNB, the minority-carrier lifetime τ_B is related to the electron concentration as [9]

$$\tau_B = (n - n_0)/U_{SRH}^{\text{SRH}} = \Delta n/U_{SRH}^{\text{SRH}} \quad (10)$$

where n_0 is the equilibrium electron concentration and Δn is the excess electron concentration. For a base with an arbitrary length [7],

$$\Delta n = \Delta n(X_2) \sinh[(X_B - x)/L_n] / \sinh[(X_B - X_2)/L_n] \quad (11)$$

Here $L_n = (D_n \tau_B)^{0.5}$ is the electron diffusion length in the QNB and, using the thermionic and tunneling mechanisms at hetero-interface and Boltzmann statistics in the QNB [10]

$$\Delta n(X_2) = qv_n \gamma_n N_E \exp(-V_{B1}/V_T)/\zeta \quad (12)$$

$$\zeta = qD_n/(X_B - X_2 + D_n/v_s) + qv_n \gamma_n \exp[(V_{B2} - \Delta E_C/q)/V_T] \quad (13)$$

where v_n is the electron thermal velocity, γ_n is the electron tunneling coefficient, N_E is the emitter doping concentration, and V_{B1} and V_{B2} are the barrier potentials on the emitter and base sides of the junction, respectively. Since τ_B and Δn are related to each other, a numerical procedure is needed to calculate U_{SRH} , and thus I_{QNB} , iteratively, provided the parameters associated with the SRH process (i.e., E_{Ti} , N_{Ti} , and N) are specified.

For the SCR, n , p , and ξ distributions in the base-side of SCR needed in (7) and (8) are given by

$$n(x) = n(X_2) \exp[-V_i(x)/V_T] \quad (14)$$

$$p(x) = p(X_2) \exp[V_i(x)/V_T] \quad (15)$$

$$\xi(x) = -(qN_B/\epsilon_B)(X_2 - x) \quad (16)$$

where V_i is the electrostatic potential (i.e., $V_i(X_2) = 0$ is chosen as the reference potential), N_B is the base doping concentration, and ϵ_B is the dielectric permittivity in base. The position-dependent V_i in the base-side of SCR can be expressed as [7]

$$V_i(x) = -0.5(qN_B/\epsilon_B)(X_2 - x)^2 \quad (17)$$

As will be shown later, I_{BASE} has an ideality factor of about 3 in the mid-voltage range and thus is the current component contributing to the abnormal base current observed in the post-burn-in HBT.

3. RESULTS AND DISCUSSIONS

We first investigate the effects of E_{Ti} and N on the recombination current in the base. The device considered has a typical make-up of $5 \times 10^{17} \text{ cm}^{-3}$ emitter doping concentration, $0.15 \mu\text{m}$ emitter layer thickness, 10^{19} cm^{-3} base doping concentration, and $0.1 \mu\text{m}$ base layer thickness. Also, the conduction band edge E_C has been chosen as the reference for E_{Ti} (i.e., $E_{Ti} = 0$ if located at E_C). Fig. 2 shows I_{BASE} calculated from the model using fixed $N_{Ti} = 10^{19} \text{ cm}^{-3}$ and a single trap with $E_{Ti} = 0.7 \text{ eV}$, a single trap with $E_{Ti} = 1.4 \text{ eV}$, and multiple traps with $E_{Ti} = 0.7, 1.1, \text{ and } 1.4 \text{ eV}$ (i.e., $N = 3$). The results suggest that I_{BASE} is relatively insensitive to E_{Ti} , but depends

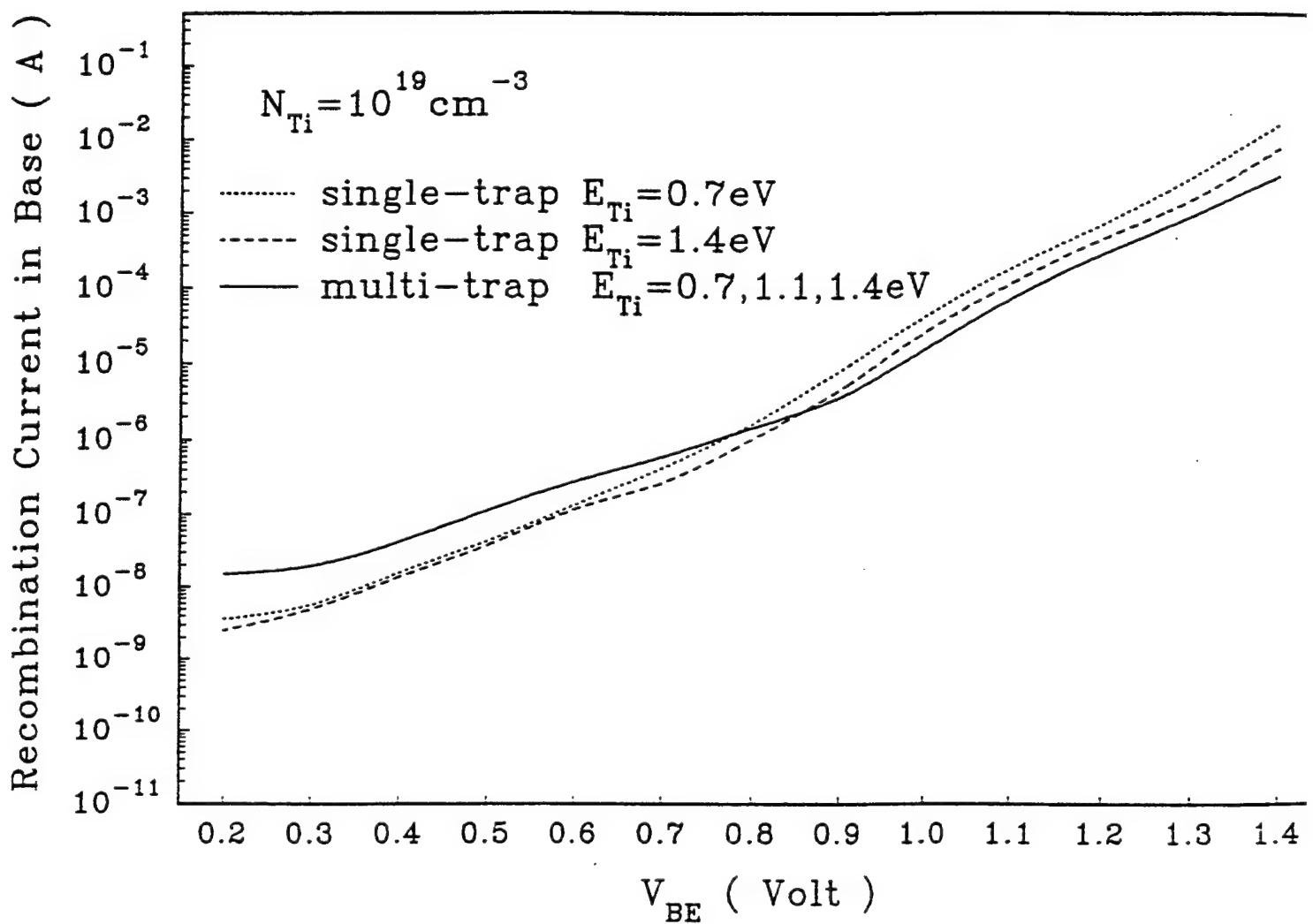


Fig. 2 Recombination current in the base vs V_{BE} calculated from the model for three different cases of N_{Ti} and N .

more on the number of trapping state N , particularly at small V_{BE} . Furthermore, all three currents exhibit an $n \approx 3$ characteristic.

Intuitively, one expects I_{BASE} increases with increasing E_{Ti} and increasing N because U^{SRH} is inversely and directly proportional $\cosh(E_{Ti}/kT)$ and N (see Eqs. (6) and (7)), respectively. This is true for small V_{BE} (i.e., $V_{BE} < 0.8$ V), where the electric field in the SCR is high, and recombination via trap-assisted tunneling in the SCR is the dominant process. For high V_{BE} , however, the electric field in the SCR is small, and the SRH recombination in the QNB is more significant. Since U^{SRH} in the QNB is a function of the electron concentration, an increase in E_{Ti} and increase in N will tend to increase U^{SRH} , but such a change will also tend to decrease τ_B and therefore decrease the electron concentration and U^{SRH} in the QNB. This compensating mechanism leads to a less significant effect of E_{Ti} and N on I_{BASE} , as observed in the region of $V_{BE} > 0.8$ V in Fig. 2. To further demonstrate this, we show in Fig. 3 I_{BASE} vs V_{BE} calculated with and without trap-assisted tunneling. It can be seen that the current component resulted from trap-assisted tunneling is negligible if V_{BE} is greater than 0.8 V. For this bias region, recombination current in the QNB is the dominant current, and I_{BASE} is less insensitive to N_{Ti} and N , as observed in Fig. 2. Also note that the abnormality of $n \approx 3$ is more evident in I_{BASE} with trap-assisted tunneling.

The dependence of $U^{SRH}(x)$ on V_{BE} is illustrated in Fig. 4, with details of $U^{SRH}(x)$ in the base-side of SCR also shown to illustrate the SRH recombination rate due to trap-assisted tunneling in the region. The recombination rate in the SCR decreases with increasing V_{BE} because of a smaller electric field and thus a smaller trap-assisted tunneling factor in the region.

Fig. 5 shows the effect of N_{Ti} on the recombination current in the base. Here, we have arbitrarily chosen a single trap with $E_{Ti} = 0.7$ eV in calculations. Clearly, the value of N_{Ti} affects I_{BASE} significantly, and N_{Ti} will be the main parameter in fitting the model calculations with experimental data.

Fig. 6 shows the total base currents of pre- and post-burn-in HBT-1 (device make-up and its leakage current parameters are given in Table I) calculated from the model and obtained from measurements. The plateau-like current for $V_{BE} < 0.8$ V in the pre-burn-in HBT is the base leakage current. For the post-stress HBT, the current behavior for $V_{BE} > 0.2$ V is changed to that of $n \approx 3$. This is due to the fact that, in addition to the base leakage current, there is a large I_{BASE} in the post-burn-in HBT. $N_{Ti} = 8.75 \times 10^{18} \text{ cm}^{-3}$ has been used to fit the model to measured data, suggesting the stress-induced defect density in such an HBT is $8.75 \times 10^{18} \text{ cm}^{-3}$. A single trap with $E_{Ti} = 0.7$ eV has also been used.

Fig. 7 shows the total base currents of pre- and post-burn-in HBT-2 (see Table I) calculated from the model and obtained from measurements [3]. For this device, we found that the burn-in test resulted in $N_{Ti} = 2 \times 10^{18} \text{ cm}^{-3}$ in the base. This is smaller than N_{Ti} in HBT-1, due perhaps to the fact that HBT-2 is subjected to a less severe burn-in test (200°C and $7 \times 10^3 \text{ A/cm}^2$) than HBT-1 (240°C and 10^4 A/cm^2).

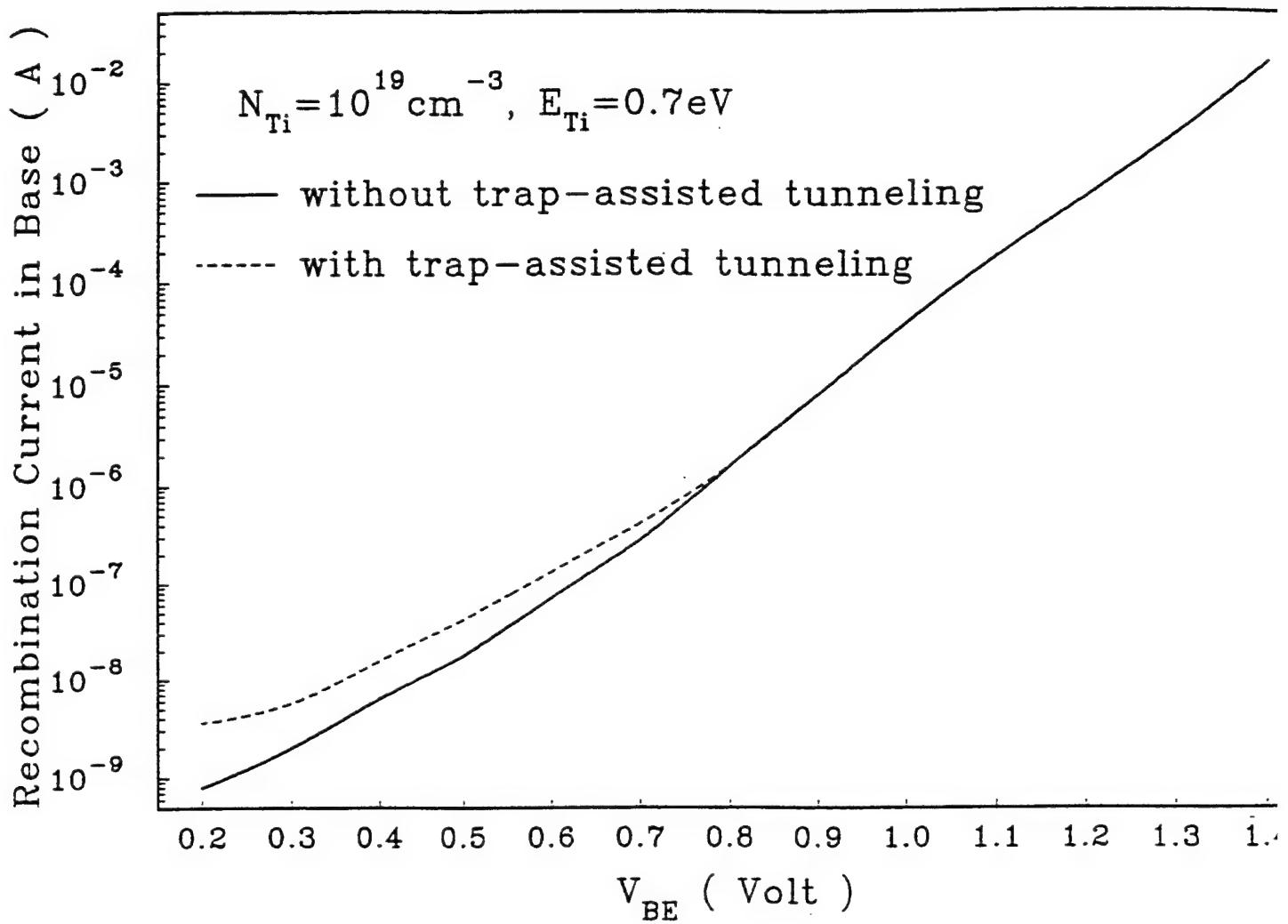


Fig. 3 Recombination current in the base calculated with and without the trap-assisted tunneling mechanism.

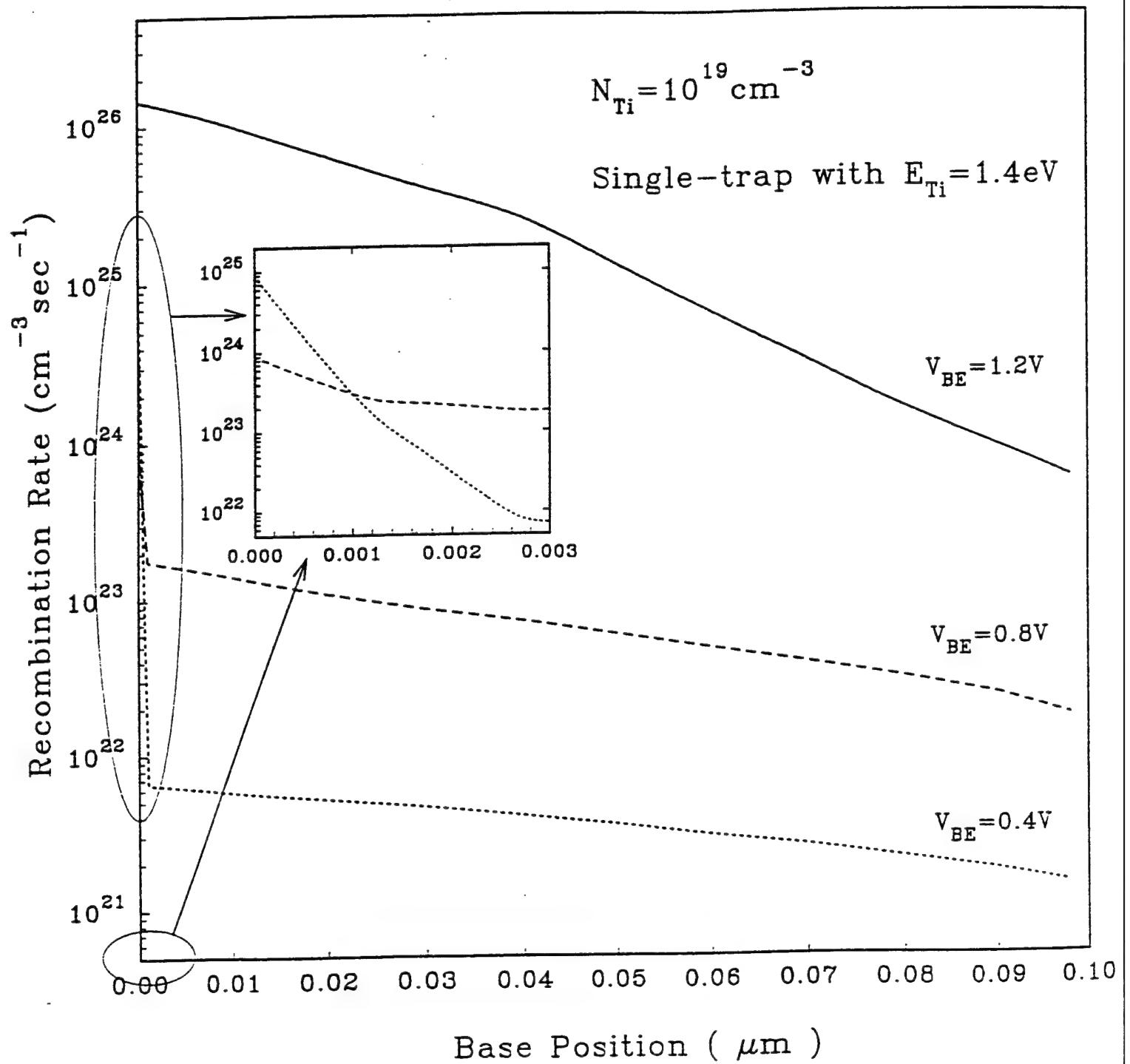


Fig. 4 SRH recombination rates vs base position calculated for three different V_{BE} . Also shown are the details of SRH recombination rates in the base-side of the SCR.

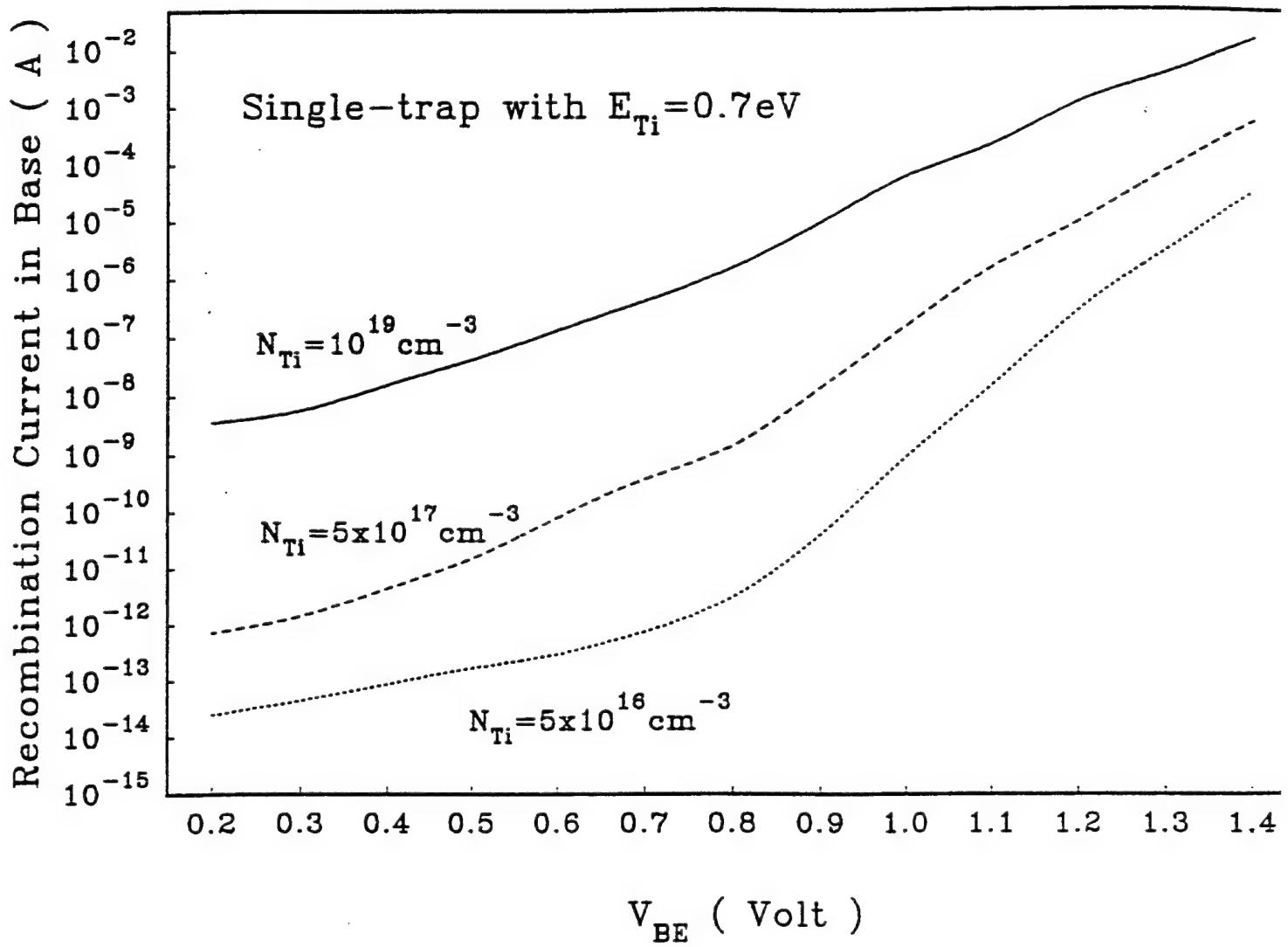


Fig. 5 Recombination current in the base vs V_{BE} calculated from the model for three different N_{Ti} .

Table I HBT Structures and Leakage Current Parameters

Parameters	HBT-1	HBT-2
Emitter doping (cm^{-3})	5×10^{17}	5×10^{17}
Emitter thickness (μm)	0.17	0.18
Emitter area (μm^2)	100	30
Base doping (cm^{-3})	1×10^{19}	1×10^{19}
Base thickness (μm)	0.1	0.14
J'_{BL} (A/cm^2)	1×10^{-5}	1.33×10^{-6}
F_L	0.005	0.005

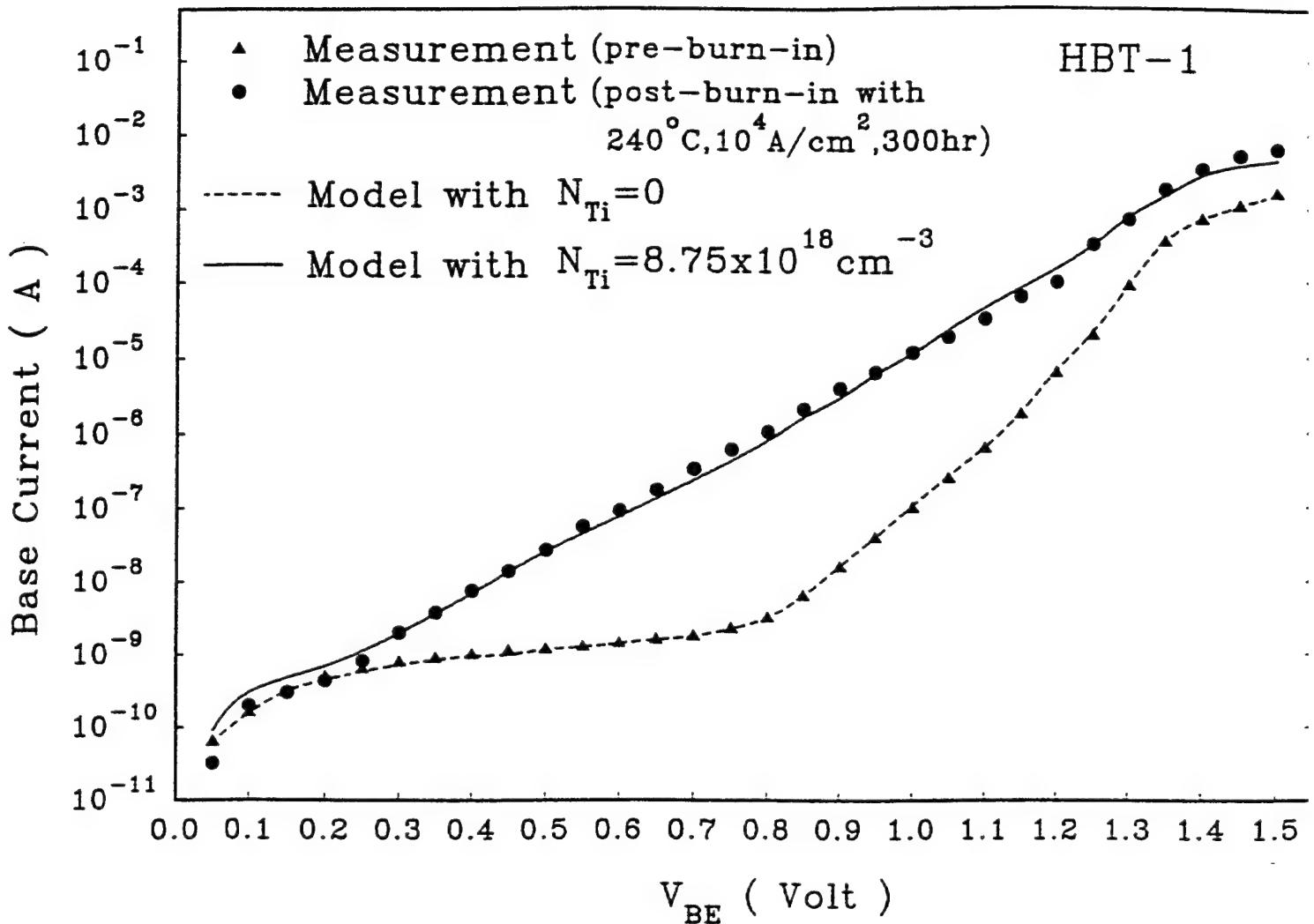


Fig. 6 Pre- and post-burn-in base currents of HBT-1 calculated from the model and obtained from measurements.

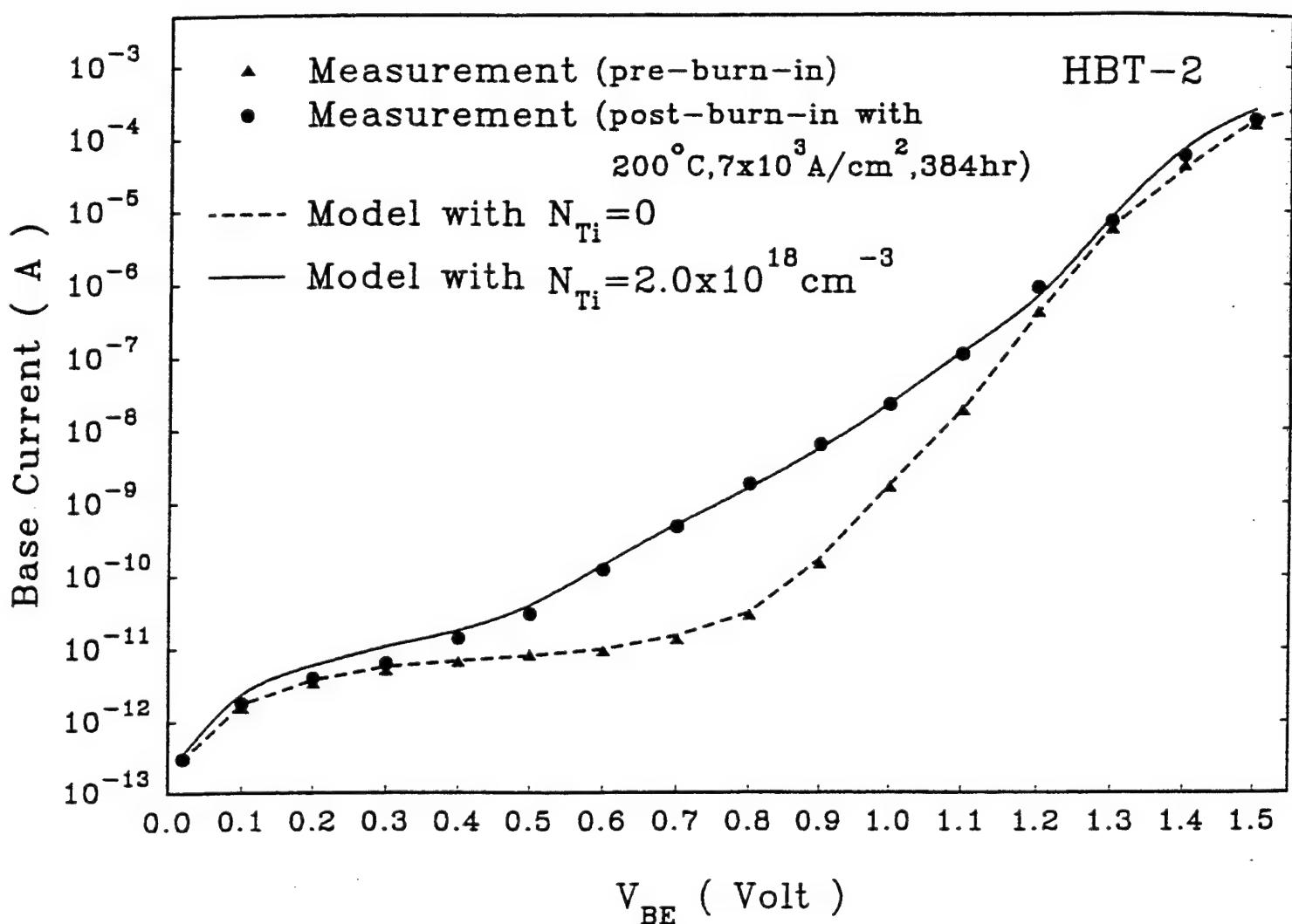


Fig. 7 Pre- and post-burn-in base currents of HBT-2 calculated from the model and obtained from measurements [3].

4. CONCLUSIONS

A model has been developed to investigate the physical mechanisms underlying the abnormal base current (i.e., with an ideality factor of about 3) observed in the post-burn-in AlGaAs/GaAs heterojunction bipolar transistor (HBT). Our study confirms the finding of recent experimental work that such a current resulted from the significant electron-hole recombination via stress-induced defect centers in the base of the HBT. Furthermore, it has been shown that the trap-assisted tunneling is an important mechanism for recombination in the space-charge region when the bias voltage is relatively low. The model calculations compare favorably with data measured from two different HBTs.

Acknowledgements--This work was supported in part by a research grant funded by the Air Force Office of Scientific Research.

References

- [1] M. E. Hafizi, L. M. Pawlowicz, L. T. Tran, D. K. Umemoto, D. C. Streit, A. K. Oki, M. E. Kim, and K. H. Yen, "Reliability analysis of GaAs/AlGaAs HBT's under forward current/temperature stress," Digest IEEE GaAs IC Symp., p. 329, 1990.
- [2] T. Henderson, D. Hill, W. Liu, D. Costa, H. F. Chau, T. S. Kim, and A. Khatibzadeh, "Characterization of bias-stressed carbon-doped GaAs/AlGaAs power heterojunction bipolar transistors," Digest IEEE IEDM, 1994.
- [3] H. Sugahara, J. Nagano, T. Nittono, and K. Ogawa, "Improved reliability of AlGaAs/GaAs heterojunction bipolar transistors with a strain-relaxed base," Digest IEEE GaAs IC Symp., p. 115, 1993.
- [4] J. J. Liou and C. I. Huang, "Base and collector currents of pre- and post-burn-in AlGaAs/GaAs heterojunction bipolar transistors," Solid-St. Electron., vol. 37, p. 1349, 1994.
- [5] J. L. Benton, M. Levinson, A. T. Macrander, H. Temkin, and L. C. Kimerling, "Recombination enhanced defect annealing in n-InP," Appl. Phys. Lett., vol. 45, p. 566, 1984.
- [6] J. J. Liou, C. I. Huang, B. Bayraktaroglu, D. C. Williamson, and K. B. Parab, "Base and collector leakage currents of AlGaAs/GaAs heterojunction bipolar transistors," J. Appl. Phys., vol. 76, p. 1349, 1994.
- [7] For example, see J. J. Liou, *Advanced Semiconductor Device Physics and Modeling*, Boston: Artech House, Inc., 1994.
- [8] G. A. M. Hurkx, D. B. M. Klaassen, and M. P. G., "A new recombination model for device simulation including tunneling," IEEE Trans. Electron Devices, vol. 39, p. 331, 1992.
- [9] Chih-Tang Sah, *Fundamentals of Solid-State Electronics*, Singapore: World Scientific, 1991, Ch. 3.
- [10] M. S. Shur, *GaAs Devices and Circuits*, New York: Plenum, 1987.

THERMOPHYSICAL INVARIANTS FROM LWIR IMAGERY FOR ATR

N. Nandakumar
Assistant Professor
Department of Electrical Engineering

University of Virginia
Charlottesville, VA 22903-2442

Final Report:

Summer Faculty Research Extension Program
Subcontract No. 95-0861
Wright Laboratory

Sponsored by:

Air Force Office of Scientific Research
Bolling Air Force Base, DC
and
Wright Laboratory

December 1995

THERMOPHYSICAL INVARIANTS FROM LWIR IMAGERY FOR ATR

N. Nandhakumar

Assistant Professor

Department of Electrical Engineering

University of Virginia

Abstract

We recently formulated a new approach for computing invariant features from infrared (IR) images. That approach is unique in the field since it considers not just surface reflection and surface geometry in the specification of invariant features, but it also takes into account internal object composition and thermal state which affect images sensed in the non-visible spectrum. In this paper we extend the thermophysical algebraic invariance (TAI) formulation for the interpretation of uncalibrated infrared imagery, and further reduce the information that is required to be known about the environment. Features are defined such that they are functions of only the thermophysical properties of the imaged objects. In addition, we show that the distribution of the TAI features can be accurately modeled by symmetric alpha-stable models. This approach is shown to yield robust classifier performance. Results on ground truth data and real infrared imagery are presented. The application of this scheme for site change detection is discussed.

THERMOPHYSICAL INVARIANTS FROM LWIR IMAGERY FOR ATR

N. Nandakumar

1 Introduction

Object recognition requires robust and stable features that are separable in feature space. An important characteristic of these features is that they be invariant to scene conditions, such as illumination, and changes in viewpoint/object pose. The formulation of invariant features, and the quantitative analysis of feature variance is currently being addressed by a number of researchers in the computer vision community, and has led to the establishment of a mature and growing theory of feature invariance. Such efforts have primarily considered reflected-light imagery – formed by sensing visible wavelength energy. This investigation has resulted in a number of distinct (yet related) approaches that may be loosely grouped into three categories: (1) Geometric Invariance (GI), (2) Quasi-Invariance (QI), and (3) Intensity-based Invariance (II).

Geometric invariants have been investigated since the inception of the field of image analysis in the early 1960s (actually, such investigation can be traced back to the onset of photogrammetry in the 19th century). The main issue is the investigation of features that are invariant with respect to changes in viewpoint. Geometric invariants come in two basic “flavors”, algebraic and differential. Algebraic invariants are based on the global configuration of features extracted from an object, and involve the notion of algebraic shapes, e.g., shapes are analytically expressed as 2D conics, and invariant relationships are identified between conics belonging to an object. Differential invariants are polynomial expressions involving the local curvature properties of 2D and 3D curves and are computed for each point on the curve. Thus differential invariants are actually parameter space “signatures” (e.g., a locus of point in an abstract parameter space) that are unique to the object, so that differences between two parameter space signatures define different objects. A close relationship (an equivalence in some cases) has been shown between some formulations of differential invariants and algebraic invariants [Forsyth et al., 1991]. There are several examples of geometric (actually algebraic) invariants of planar configurations under projective transformation, such as the cross ratio using 4 collinear points, 5 coplanar points with no three being collinear, a conic and two non-tangent lines, and a pair of coplanar conics. More recently, an invariant of a pair of non-coplanar conics has been proposed [Long, 1995].

Quasi-Invariance (QI) can be thought of as a relaxation of the central notion of GI [Binford et al., 1989, Zerroug and Nevatia, 1993]. A Quasi-Invariant is a property of a geometric configuration that is almost invariant under a class of imaging transformations. Formally, a geometric configuration is a QI if the linear term(s) in the Taylor series expansion of the configuration with respect to the parameters of the imaging transformation vanish. This has the effect of making the QI measure nearly constant over a large region of

the viewing hemisphere, and rapidly diverging only as the angle between the view direction and the surface normal approaches large values. This behavior lends itself to probabilistic modeling and the use of reasoning schemes such as Bayesian evidence accrual. A detailed study of the variation of "invariant" features with viewpoint has been undertaken [Burns et al., 1993].

Intensity-based Invariants (II's) are functions of image intensities that yield values which are invariant to scene illumination and viewpoint. To date, some investigation of II's has been conducted for visible imagery, but practically none has been reported for non-visible imagery. Examples of II's in computer vision include color features for object recognition [Klinker et al., 1988, Healey, 1991, Healey and Slater, 1994] and polarization cues for material identification [Wolf, 1990]. A more direct example of this approach computes ratios of albedos of homogeneous image intensity patches within objects in visible imagery [Nayar and Bolle, 1993].

Non-visible modalities of sensing have been shown to greatly increase the amount of information that can be used for object recognition. A very popular and increasingly affordable sensor modality is thermal imaging - where non-visible radiation is sensed in the long-wave infrared (LWIR) spectrum of $8\mu m$ to $14\mu m$. The current generation of LWIR sensors produce images of contrast and resolution that compare favorably with broadcast television quality visible light imagery. However, the images are no longer functions of only surface reflectance. As the wavelength of the sensor transducer passband increases, emissive effects begin to emerge as the dominant mode of electromagnetic energy exitance from object surfaces. The (primarily) emitted radiosity of LWIR energy has a strong dependence on internal composition, properties, and state of the object such as specific heat, density, volume, heat generation rate of internal sources, etc. This dependence may be exploited by specifying image-derived invariants that vary only if these parameters of the physical properties vary.

The derivation of thermophysical invariants (TI's) from non-visible wavelength imagery, the evaluation of the performance of these invariants, and their use in object recognition systems poses several advantages. The main advantage of this approach is the potential availability of a number of new (functionally independent) invariants that depend on internal compositional properties of the imaged objects. Note that it is possible to evaluate the behavior of thermophysical invariants using ground truth data consisting of images of objects of known composition and internal state. This additional information can be used to augment/complement the behavior of GI's. One way in which GI's can be integrated with TI's for object recognition is as follows: (1) Parametric curves and/or lines are extracted from an LWIR image. (2) The curves are used to compute GI's which are in turn used to hypothesize object identity and pose, and (3) TI's are computed for this hypothesis and compared to a stored model library for verification. Some details of this approach are presented later.

In this paper we first review a recently reported scheme that establishes thermophysical quasi-

invariants from LWIR imagery. This scheme uses the principle of conservation of energy at the surface of the imaged object to specify a functional relationship between the object's thermophysical properties (e.g., thermal conductivity, thermal capacitance, emissivity, etc.), scene parameters (e.g., wind temperature, wind speed, solar insolation), and the sensed LWIR image gray level. We use this functional form to derive features that remain relatively constant despite changes in scene parameters/driving conditions. In this formulation the internal thermophysical properties play a role that is analogous to the role of parameters of the conics, lines and/or points that are used for specifying geometric invariants when analyzing visible wavelength imagery. Thus, in addition to the currently available techniques of formulating features that depend only on external shape and surface reflectance discontinuities, the phenomenology of LWIR image generation can be used to establish new features that "uncover" the composition and thermal state of the object, and which do not depend on surface reflectance characteristics. However, this previous technique requires a great deal of scene information to derive the thermophyscial features, and also requires radiometrically calibrated imagery.

Next, we extend an existing scheme for forming thermophysical invariant features by using principles of algebraic invariance theory [Hilbert, 1887]. The previous TAI approach depended on the use of radiometrically calibrated imagery, i.e. imagery where the relationship between the sensed gray scale values and the actual temperature of the imaged object is assumed known [Nandhakumar, et al, 1995]. Since radiometrically calibrated LWIR imagery is not always available and in many cases such calibration is not possible, the previously reported interpretation algorithms are limited to a relatively small range of applications. The new approach presented in this paper allows the use of uncalibrated imagery, thus broadening the application of the overall approach. Furthermore, unlike the previous approach the new approach does not require any knowledge of the ambient conditions of the scene. It accounts for, and is tolerant to, variation in atmospheric attenuation from scene to scene, and also missing/unavailable measurement of the ambient temperature at the scene. This further detaches the object recognition process from the environmental conditions under which the object is imaged, and by requiring less *a priori* knowledge of the scene, the dependence of the feature on the intrinsic properties of the object is increased. The new TI approach is used to generate features in a model-based object recognition scheme. The results are promising in that the features derived show strong intra-class stability - the feature value is constant for a given object class and under varying conditions. The features also exhibit good separability characteristics in inter-class tests - the feature values extracted from other object classes are well separated. In addition, we show that the distribution of the TAI features can be accurately modeled by symmetric alpha-stable models. This approach is shown to yield robust classifier performance.

Another computer vision/image understanding task that is closely related to object recognition is the task of automatic site change detection. In one example of such a task, images are obtained, periodically, by an aircraft or satellite flying over a site to be monitored. The imagery is compared with known prior

information or detailed site models to determine if any changes have occurred. For example, it may be important to detect if a patch of gravel or dirt has been replaced with a concrete or asphalt surface at some factory or construction site being monitored. Since some information is usually available of the site being monitored (in the form of site models), and also of the imaging parameters of the sensors, the site change detection task follows the paradigms of context-based vision and model-based vision. It is also closely related to the task of object recognition where a hypothesis of an object composed of different material types is verified or refuted. A particular site may be considered to be a composition of a specific collection of materials. A feature may be used to verify the existence of this composition. A change in the site should result in a feature value other than that expected for the original site. We discuss the application of our approach to site change detection and present results of analyzing real LWIR imagery.

The ideas presented in this paper are continuations and extensions of previous and ongoing research in thermophysical model-based interpretation of LWIR imagery. Section 2 describes relevant past work on thermophysical approaches for interpreting infrared imagery. A description of using principles of algebraic invariance to formulate thermophysical invariants is presented in section 3, followed by extensions to deal with radiometrically uncalibrated IR imagery. Preliminary experimental results of applying this new approach to real imagery are presented in section 4, which is followed by a discussion of the behavior of the new method, statistical models for the distribution of the new features, issues to be considered in using this method for object recognition, and issues that remain to be explored.

2 Past Thermophysical Approaches to IR Image Analysis

2.1 Heuristic Methods

An intuitive approach to thermophysical interpretation of LWIR imagery is given in [Gauder, et al, 1993]. This approach rests upon the following observation, termed the “Thermal History Consistency Constraint” and analogous to Lowe’s well known Viewpoint Consistently Constraint [Lowe, 1987]: “The temperature of all object features for a passive object must be consistent with the heat flux transfer resulting from exposure to the same thermal history.” In [Gauder, et al, 1993] this constraint is exploited by analyzing objects to locate components that are similar in terms of thermophysical properties and then examining a temporal sequence of calibrated LWIR data to experimentally assess the degree to which such thermophysically similar components exhibit similar temperature state temporal behavior. Such analysis was shown to lead to formulation of simple intensity ratio features. No experimentation was done with multiple objects to examine between and within-class separation, so little can be drawn in the way of a substantive conclusion with respect to utility as an object identification technique.

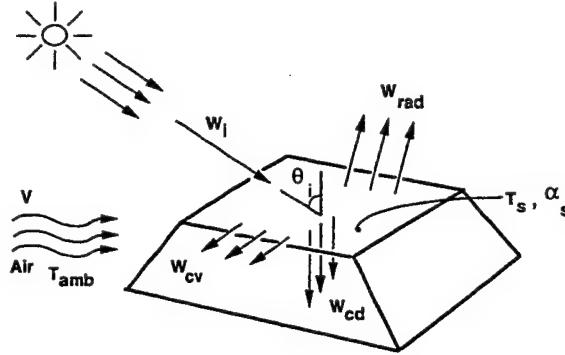


Figure 1: Energy exchange at the surface of the imaged object. Incident energy is primarily in the visible spectrum. Surfaces loses energy by convection to air, via radiation to the atmosphere, and via conduction to the interior of the object.

2.2 A Physics-Based Model

A physics-based approach has been attempted to establish invariant and quasi-invariant features which depend only on thermophysical object properties. Such an approach requires that a model be developed based on the principle of the conservation of energy at an elemental surface patch on the object. An overview of this approach is presented below. The use of this model in a previously reported technique is then presented and a new approach that uses algebraic invariance theory on this model is described in section 3.

At the surface of the imaged object (figure 1), energy absorbed by the surface equals the energy “lost” to the environment and to the interior of the object:

$$W_{abs} = W_{lost} \quad (1)$$

Energy absorbed by the surface is given by

$$W_{abs} = W_I \cos\theta_I \alpha_s, \quad (2)$$

where, W_I is the incident solar irradiation on a horizontal surface and θ_I is the angle between the direction of irradiation and the surface normal, and α_s is the surface absorptivity which is related to the visual reflectance ρ , by $\alpha_s = 1 - \rho$. The energy “lost” by the surface was given by

$$W_{lost} = W_{cnd} + W_{st} + W_{cv} + W_{rad}, \quad (3)$$

where W_{cnd} denotes the energy conducted from the surface into the interior of the object, W_{st} is the energy stored, W_{cv} denotes the energy convected from the surface to the air which has temperature T_{amb} and velocity V , and W_{rad} is the energy lost by the surface to the environment via radiation, The radiation energy loss is computed from:

$$W_{rad} = \epsilon \sigma (T_s^4 - T_{amb}^4), \quad (4)$$

where, σ denotes the Stefan-Boltzman constant, T_s is the surface temperature of the imaged object, and T_{amb} is the ambient temperature.

The convected energy transfer is given by

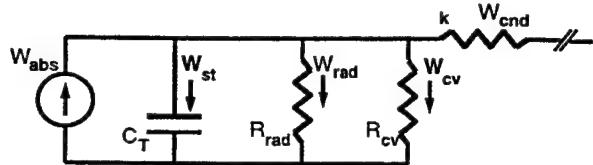


Figure 2: The equivalent thermal circuit for the extended model that separates the stored energy component and the conduction component to the interior of the object.

$$W_{cv} = h(T_s - T_{amb}) \quad (5)$$

where, h is the average convected energy transfer coefficient for the imaged surface, which depends on the wind speed, thermophysical properties of the air, and surface geometry [Incropera and DeWitt, 1981].

The conducted energy flow is decomposed into two terms W_{cnd} , which is the energy conducted to the interior of the object, and W_{st} which is a stored energy term.

$$W_{cnd} = -k dT/dx, \quad (6)$$

where k is the thermal conductivity of the material, and x is distance below the surface. Here, we assume that lateral energy conduction is insignificant compared to conduction along the direction normal to the surface. The stored energy term is

$$W_{st} = C_T \frac{dT_s}{dt} \quad (7)$$

where C_T is the lumped thermal capacitance of the object and is given by

$$C_T = DVc$$

where, D is the density of the object, V is the volume, and c is the specific heat. Figure 2 shows the equivalent thermal circuit for this energy flow model. The resistances are given by:

$$R_{cv} = \frac{1}{h} \quad \text{and} \quad R_{rad} = \frac{1}{\epsilon\sigma(T_s^2 + T_{amb}^2)(T_s + T_{amb})} \quad (8)$$

2.3 Establishing a Quasi-Invariant

The physics-based model described above was used to interpret registered IR and visual imagery from an outdoor scene [Nandhakumar & Aggarwal 1988a]. First, a low Biot number was assumed, viz., the surface was considered to be a thin plate, hence $W_{cnd} = 0$. A simplified shape-from-shading approach was used to compute $\cos\theta_I$ and α_s from the visual image. The surface temperature, T_s , was estimated from the thermal image based on an appropriate model of radiation energy exchange between the surface and the infrared camera. W_I is given by available empirical models (based on time, date and latitude of the scene) or by measurement with a pyranometer. Note that it is reasonable to use the visual reflectance to estimate the energy absorbed by the surface since approximately 90% of the energy in solar irradiation lies in the visible wavelengths [Incropera and DeWitt, 1981]. The wind speed and ambient temperature are also measured for the scene. Thus the energy flows, W_{abs} , W_{cv} , W_{rad} and hence W_{st} may be computed from the image and scene information.

It is clear from figure 2 that the conduction energy term W_{st} depends on the lumped thermal capacitance C_T of the object and can be used to describe the object's ability to sink/source thermal radiation, a feature shown to be useful in discriminating between different classes of objects. In order to minimize the feature's dependence on differences in absorbed energy flux, a normalized feature was defined to be the ratio $R = W_{st}/W_{abs}$. The values were found to be lowest for vehicles, highest for vegetation and in between for buildings and pavements. Classification of objects using this property value is discussed in [Nandhakumar and Aggarwal, 1988b].

Although this approach for integrated analysis of thermal and visual imagery is powerful in that it makes available features that are completely defined by internal object properties, the thermophysical feature estimates were not sufficiently reliable due to errors in segmentation and registration of the thermal and visual image pairs, and the noise-sensitivity of shape-from-shading techniques used in computing relative orientation of the surface [Zhang, et al 1994]. A statistically robust scheme for computing the thermophysical feature R was proposed to minimize this drawback [Nandhakumar, 1994]. However, the computational complexity for such a technique is very high. Another limitation in the previous formulation is that a low Biot number was assumed, viz., the surface was assumed to be a thin plate, which is rarely satisfied in practise. Also, the technique requires *a priori* knowledge of several surface and scene parameters such as emissivity, wind speed, solar insolation, etc, which in many applications are unavailable. Even in those situations where such information is available, the thermophysical feature, R , is only weakly invariant. While separation between classes is preserved, the range of values of R for each class is observed to vary with time of day and season of year. Also, the feature R is able to only separate very broad categories of objects, such as automobiles, buildings, and vegetation - it lacks the specificity to differentiate between different models of vehicles.

An improved formulation that attempts to overcome these limitations is described in section 3, wherein the feature is constrained to be invariant to affine transformations of the driving (scene) conditions.

3 Thermophysical Algebraic Invariants (TAI's)

An improved approach for computing invariant features that depend on the internal, thermophysical properties of the imaged object, and that are invariant to driving (scene and surface) parameters is based on a reformulation of the model outlined in section 2. In this formulation we explicitly eliminate the requirement that the ambient temperature be known, and we eliminate the requirement that radiometric calibration be known. First, to account for reasonable values of Biot numbers, viz. for objects other than thin plates, W_{cnd} is no longer assumed to be zero. Next, the radiosity at the surface of an object can be approximated to be linearly dependent on surface temperature. Consider equation (4) for the radiation energy loss term. A standard simplification of this expression results in the following form [Incropera and DeWitt, 1981],

$$W_{rad} = \epsilon \sigma (T_s + T_{amb})(T_s^2 - T_{amb}^2)(T_s - T_{amb}) = h'(T_s - T_{amb}), \quad (9)$$

where h' is assumed to be constant over a specific range of temperatures. Note that when the variation in scene temperature is no more than $\pm 50^\circ C$ the error due to the linearity assumption is less than 10% (figure 3), which we have found to be comparable to the accuracy possible by radiometric calibration.

This observation also implies that the gray level, L_s , of the LWIR image is related to the surface temperature, T_s , of the object's surface by a linear relationship,

$$L_s = \frac{1}{\beta}(T_s + \mu). \quad (10)$$

Using the above approximations the energy balance equation, $W_{abs} = W_{rad} + W_{cv} + W_{st} + W_{cnd}$ may be now rewritten in the following linear form:¹

$$a^1 x_1 + a^2 x_2 + a^3 x_3 + a^4 x_4 + a^5 x_5 = 0. \quad (11)$$

where

$$\begin{aligned} a^1 &= \cos\theta_I & x_1 &= W_I \alpha_s \\ a^2 &= L_s & x_2 &= -\beta(h + h') \\ a^3 &= C_T & x_3 &= -\frac{dT_s}{dt} \\ a^4 &= k & x_4 &= -T_s + T_{int} \\ a^5 &= 1 & x_5 &= -(T_{amb} + \mu)(h + h') \end{aligned} \quad (12)$$

Any pixel in the LWIR image of an object will yield a 5-D measurement tensor, \mathbf{a} . The image measurement (gray level) specifies a^2 . The values for a^1 , a^3 , and a^4 are known when the identity and pose of the object are hypothesized. The "driving conditions", or unknown scene parameters that change from scene to scene are described by x_i . For each pixel in the thermal image eqn (12) defines a hyperplane in 5-D space. Note that the last element of the measurement tensor is always unity.

3.1 An Algebraic Invariance Formulation

Consider two different LWIR images of a scene obtained under different scene conditions and from different viewpoints. Consider N points on the object that (a) are visible in both views, and (b) have been selected to lie on different components of the object which differ in material composition and/or surface normal direction. Assume (for the nonce) that the object pose for each view, and point correspondence between the two views are available (or hypothesized). A point in each view yields a contravariant tensor a^i as defined by eqn (12). Let the collection of these tensors be denoted by a_k^i , $k = 1, 2, \dots, N$ for the first scene/image and b_k^i , $k = 1, 2, \dots, N$ for the second scene. For the k -th point we denote the measurement tensor as \mathbf{a}_k for the first view, and as \mathbf{b}_k for the second view, and the driving conditions tensor as \mathbf{x}^k .

¹Here, we use the Einstein notation to denote the image-based measurement vector as a contravariant tensor, a^i , while the changing scene conditions form a covariant tensor, x_i . Therefore, $a^i x_i = 0$.

Figure 4: We need a collection of five points in 5D measurement space to compute the affine transformation, T_i , between two scenes. Two different sets of 5 points each can be used to define an absolute invariant provided $\det(T_1) = \det(T_2)$.

We assume that the scene/driving conditions, \mathbf{x}^k for the first view and \mathbf{y}^k for the second view, are related by an affine transformation. The justification of this assumption is discussed below. We have found, empirically, that this assumption holds when the points are selected using the method discussed later in this paper. Since the \mathbf{x}^k are transformed affinely, then it follows that the \mathbf{a}_k are also transformed affinely. Note that an affine transformation from one scene to another is trivial to obtain if we have only five points that generate five non-coplanar tensors in our 5-D measurement space. Consider one such subset of 5 of the N points, and denote them as h, j, l, m , and n .

The determinant

$$d(\mathbf{a}_h \mathbf{a}_j \mathbf{a}_l \mathbf{a}_m \mathbf{a}_n) = \begin{vmatrix} a_h^1 & a_h^2 & a_h^3 & a_h^4 & a_h^5 \\ a_j^1 & a_j^2 & a_j^3 & a_j^4 & a_j^5 \\ a_l^1 & a_l^2 & a_l^3 & a_l^4 & a_l^5 \\ a_m^1 & a_m^2 & a_m^3 & a_m^4 & a_m^5 \\ a_n^1 & a_n^2 & a_n^3 & a_n^4 & a_n^5 \end{vmatrix} = |A| \quad (13)$$

defines the “volume” of the oriented parallelepiped formed by the pencil of the five contravariant tensors $\mathbf{a}_h, \mathbf{a}_j, \mathbf{a}_l, \mathbf{a}_m, \mathbf{a}_n$. The above determinant is a relative invariant to the affine transformation [Gurevich, 1964], i.e.,

$$d(\mathbf{a}_h \mathbf{a}_j \mathbf{a}_l \mathbf{a}_m \mathbf{a}_n) = \delta_{hjlmn} \times d(\mathbf{b}_h \mathbf{b}_j \mathbf{b}_l \mathbf{b}_m \mathbf{b}_n) \quad (14)$$

where δ_{hjlmn} is the determinant of the affine transformation, T_{hjlmn} , which relates the measurement tensors, i.e., $\mathbf{a}_k = \mathbf{b}_k T_{hjlmn}$, $k \in \{h, j, l, m, n\}$.

Consider another set of five points in which at least one point is different from the previous set. Denote this second set as $\{p, q, r, s, t\}$. Again, assume that the measurement tensors for this collection of points undergo an affine transformation from the first scene to the second, and denote this transformation by T_{pqrs} .

$$d(\mathbf{a}_p \mathbf{a}_q \mathbf{a}_r \mathbf{a}_s \mathbf{a}_t) = \delta_{pqrs} \times d(\mathbf{b}_p \mathbf{b}_q \mathbf{b}_r \mathbf{b}_s \mathbf{b}_t) \quad (15)$$

where $\delta_{pqrs} = \det(T_{pqrs})$. Hence, if $\delta_{hjlmn} = \delta_{pqrs}$, then we can define an absolute invariant as

$$I = \frac{d(\mathbf{a}_h \mathbf{a}_j \mathbf{a}_l \mathbf{a}_m \mathbf{a}_n)}{d(\mathbf{a}_p \mathbf{a}_q \mathbf{a}_r \mathbf{a}_s \mathbf{a}_t)} = \frac{|A|}{|\hat{A}|} \quad (16)$$

where A and \hat{A} are measurement matrices formed by the tensors indicated above.

Note that the existence of affine transformations T_{hjlmn} and T_{pqrs} is easy to ensure by selecting the points that lie on different material types and/or have different surface normals. However, the existence and selection of two sets of five points such that $\delta_{hjlmn} = \delta_{pqrs}$ holds is crucial to the existence and determination of an absolute invariant. Our experimental investigation shows that point sets that satisfy this equivalence class of transformations are plentiful on any real, complex object such as a vehicle. The thermophysical justification for the existence of this equivalence class is being addressed in our ongoing work.

The selection of the two sets (with five points in each) that satisfy this equivalence relationship may be attempted as a data-driven training task as follows. LWIR imagery from different object classes are

obtained at different times of day and different seasons of the year. N points are picked on an object – on distinctive components that differ in material composition and/or surface normal. Consider the image from time t_u and the image from t_v , $u \neq v$. The measurements at t_u along with the hypothesis of the identity of the object form the tensors \mathbf{a}_k . Similarly, image information at time t_v is used to form the measurement tensors \mathbf{b}_k .

All combinations of two sets of five points each, $\{h, j, l, m, n\}$ and $\{p, q, r, s, t\}$, are examined. The measurement matrices $(\mathbf{a_h a_j a_l a_m a_n})$, $(\mathbf{b_h b_j b_l b_m b_n})$, $(\mathbf{a_p a_q a_r a_s a_t})$, and $(\mathbf{b_p b_q b_r b_s b_t})$ are constructed. The transformations T_{hjlmn} and T_{pqrs} , if they exist, and their determinants δ_{hjlmn} , and δ_{pqrs} are computed. The two sets that best satisfy $\delta_{hjlmn} = \delta_{pqrs}$ for different choices of pairs of scenes/images, i.e. different choices of t_u and t_v , are selected. With N points, the number of possible choices of the pair of sets of points is given by

$$n_N = \frac{1}{2} \left(\frac{N!}{(N-5)!5!} \right) \left(\frac{N!}{(N-5)!5!} - 1 \right) \quad (17)$$

In order for the invariant feature to be useful for object recognition the value of I must be different if the measurement vector is obtained from a scene that does not contain the hypothesized object and/or the hypothesized pose is incorrect. A search for the best sets of points that both identify the object and separate the classes must be conducted over the n_N point sets. The point sets may be rated for their intra-class invariance, then the best choices may be used to evaluate their inter-class separability.

3.2 Employing TAI's for Object Recognition / Site Change Detection

The feature computation scheme formulated above is suitable for use in an object recognition system that employs a hypothesize-and-verify strategy. The scheme would consist of the following steps:

1. extract geometric features, e.g., lines and conics.
2. for image region, r , hypothesize object class, k , and pose using, for example, geometric invariants as proposed by Forsyth, et al [Forsyth, et al, 1991],
3. use the model of object k and project visible points labeled $i = 1, 2, \dots$ onto image region r using scaled orthographic projection,
4. for point labeled i in the image region, assign thermophysical properties of point labeled i in the model of object k ,
5. use the gray levels at each point and the assigned thermophysical properties, to compute the measurement matrices A and \hat{A} , and hence compute the feature $f^k(r) = |A|/|\hat{A}|$, and finally,
6. compare feature $f^k(r)$ with model prototype F_k to verify the hypothesis.

The application of this scheme for site change detection is straightforward. For a site being monitored, M different types of surfaces are selected *a priori* to produce a thermophysical affine invariant. Note that we must have $M > 5$. One may also be able to specify more than one TAI, and hence establish a feature vector. An LWIR image is first registered with the site using established techniques [Barrett, et al, 1994]. The gray levels from the M selected regions along with the known material properties are used to generate the TAI features. When one or more of the surfaces change (e.g., from gravel to concrete) then the feature (vector) computed from the scene under the hypothesis of the prior material types will produce a value different from that expected. The detection of the change is thus linked to the refutation of an incorrect hypothesis.

3.3 Reduced Dimension Forms

The linear form, eqn (11), must be slightly modified to facilitate feature specification in scenarios where the feature described above becomes undefined. Two such cases are (1) interpreting LWIR imagery acquired at night and (2) imaging an object on which all of the surface normals are parallel. For the nighttime case the solar insolation is nonexistent, W_{ab} , as defined above is zero, and the energy balance model has only four terms. Hence, the measurement tensor is four dimensional. In this case we can simply consider sets of four points in evaluating the absolute invariant. The case where all of the visible object surfaces are oriented such that their normals are parallel is slightly more complex and requires consideration of the algebraic form.

Consider two sets of points from the same scene that are used to form the feature in eqn (16). As described above, the balance of energy equation must be satisfied at each of these points. The form of an equation for a point in the first set is

$$a_k^i x_i^k = 0, \quad i = 1 \dots 5, \quad k \in \{h, j, l, m, n\} \quad (18)$$

where the variable labeling is the same as that in eqn (12). Note that $a^5 = \cos \theta$ and $x_5 = W_I \alpha_s$. Similarly the form for a point in the second set is

$$\hat{a}_k^i \hat{x}_i^{k'} = 0, \quad i = 1 \dots 5, \quad k' \in \{p, q, r, s, t\} \quad (19)$$

where again the variable labeling is from eqn (12) and $\hat{a}^5 = \cos \theta$ and $\hat{x}_5 = W_I \hat{\alpha}_s$. Since the surface normals are parallel, $\cos \theta$ is the same for all points in the two sets that form the measurement matrices, A and \hat{A} . This will cause two of the columns of the measurement matrix of eqn (13) to be linearly dependent, and the feature value will be undefined in the original formulation eqn (16). Note also that the incident solar flux is also the same for all points in the two sets.

Now consider the eqns (18) and (19) expressed with the cosine terms separated out. The equation for the first set becomes,

$$a_k^i x_i^k + \cos \theta W_I \alpha_s^k = 0, \quad i = 1 \dots 4, \quad k \in \{h, j, l, m\}. \quad (20)$$

The second set is likewise defined,

$$\hat{a}_k^i \hat{x}_i^{k'} + \cos\theta W_I \hat{\alpha}_s^{k'} = 0, \quad i = 1 \dots 4, \quad k' \in \{p, q, r, s\} \quad (21)$$

Subtracting eqns (21) from (20) yields,

$$a_k^i x_i^k = \hat{a}_k^i \hat{x}_i^{k'} + \Phi_{(k,k')} \quad i = 1 \dots 4, \quad (k, k') \in \{(h,p), (j,q), (l,r), (m,s)\} \quad (22)$$

where $\Phi = \cos\theta W_I (\alpha_s^k - \hat{\alpha}_s^{k'})$, $(k, k') \in \{(h,p), (j,q), (l,r), (m,s)\}$.

Consider the construction of the measurement matrices \tilde{A} and \hat{A} as described before in eqn (13), but which are now of size 4×4 and again are of full rank. The driving conditions tensors x_i and \hat{x}_i are used to construct the driving conditions matrices, \tilde{X} and \hat{X} where each point specifies a column vector. The matrices, \tilde{A} , \hat{A} , \tilde{X} , and \hat{X} span $\mathbb{R}^{4 \times 4}$. The driving condition matrices, \tilde{X} and \hat{X} can be related by an affine transformation, and hence the measurement matrices \tilde{A} and \hat{A} are related by an affine transformation. In order to express the transformation from one set of points to the other in a linear form, the measurement matrices and driving condition matrices are expressed in homogeneous coordinates with the translation, Φ , augmenting the driving conditions matrix. Denote the homogeneous versions of the measurement and driving condition matrices as \tilde{A}_ψ , \hat{A}_ψ , \tilde{X}_{psi} , and \hat{X}_ψ . The above relationships can be expressed in homogeneous coordinates as follows:

$$= \begin{pmatrix} a_h^1 & a_h^2 & a_h^3 & a_h^4 & 1 \\ a_j^1 & a_j^2 & a_j^3 & a_j^4 & 1 \\ a_l^1 & a_l^2 & a_l^3 & a_l^4 & 1 \\ a_m^1 & a_m^2 & a_m^3 & a_m^4 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1^h & x_1^j & x_1^l & x_1^m & 0 \\ x_2^h & x_2^j & x_2^l & x_2^m & 0 \\ x_3^h & x_3^j & x_3^l & x_3^m & 0 \\ x_4^h & x_4^j & x_4^l & x_4^m & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\ = \begin{pmatrix} a'_h^1 & \hat{a}_h^2 & \hat{a}_h^3 & \hat{a}_h^4 & 1 \\ \hat{a}_j^1 & \hat{a}_j^2 & \hat{a}_j^3 & \hat{a}_j^4 & 1 \\ \hat{a}_l^1 & \hat{a}_l^2 & \hat{a}_l^3 & \hat{a}_l^4 & 1 \\ \hat{a}_m^1 & \hat{a}_m^2 & \hat{a}_m^3 & \hat{a}_m^4 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{x}_1^h & \hat{x}_1^j & \hat{x}_1^l & \hat{x}_1^m & 0 \\ \hat{x}_2^h & \hat{x}_2^j & \hat{x}_2^l & \hat{x}_2^m & 0 \\ \hat{x}_3^h & \hat{x}_3^j & \hat{x}_3^l & \hat{x}_3^m & 0 \\ \hat{x}_4^h & \hat{x}_4^j & \hat{x}_4^l & \hat{x}_4^m & 0 \\ \Phi_h & \Phi_j & \Phi_l & \Phi_m & 1 \end{pmatrix}. \quad (23)$$

Representing the general linear transformation in homogeneous coordinates as $\tilde{X} = M \hat{X}$. Substituting the transformed variable

$$\tilde{A}_\psi \tilde{X}_\psi = \hat{A}_\psi M \hat{X}_\psi. \quad (24)$$

As in section 3.1 an absolute invariant is found when the following ratio remains constant from scene to scene

$$\frac{|\tilde{A}_\psi|}{|\hat{A}_\psi|} = |M|. \quad (25)$$

Since the augmentation of the \tilde{A} and \hat{A} into homogeneous coordinates does not affect the value of the determinant then

$$\frac{|\tilde{A}|}{|\tilde{\tilde{A}}|} = \frac{|\tilde{A}_\psi|}{|\tilde{\tilde{A}}_\psi|} = |M|. \quad (26)$$

Thus we have shown that in the case where the visible surfaces of the imaged object has parallel normals an invariant feature can be derived by modeling the driving conditions transformation as a affine transformation. In either of the two cases of analyzing nighttime imagery or objects with parallel normals, a separate training phase is required for the specification of the invariant feature. In general, the choice of points for the reduced forms will be distinct from that of the full linear form.

4 Experimental Results

4.1 Object Recognition using TAIs

The method of computing thermophysical affine invariants discussed above was applied to real LWIR imagery acquired at different times of the day. Four types of vehicles were imaged: a van, a truck, a tank, and a car (figures 5). Several points were selected (as indicated in the figures) on the surfaces of different materials and/or orientation. The measurement tensor given by eqn (12) was computed for each point, for each image/scene. The method used to select optimal sets of points $\{i, j, l, m, n\}$ and $\{p, q, r, s, t\}$ was similar to that described in section 3 – however, instead of using the equivalence of the determinants of the two affine transformations as the selection criterion, we used the variance in the values of the feature computed for different scenes containing the object, i.e., images obtained at different times of the day. Many different pairs of five-point-sets yielded features with low variance from scene to scene.

As mentioned in section 3 one must consider inter-class behavior as well as intra-class behavior. To investigate this we adopted the following procedure. Given an image of a vehicle, (1) assume the pose of the vehicle is known, then (2) use the front and rear wheels to establish an object centered reference frame. The center of the wheel is used as the origin, and center of the front wheel is used to specify the direction and scaling of the axes. The coordinates of the selected points are expressed in terms of this 2D object centered frame. For example, when a van vehicle is hypothesized for an image actually obtained of a car or some unknown vehicle, the material properties of the van are used, but image measurements are obtained from the image of the car at locations given by transforming the coordinates of the van points (in the van center coordinate frame) to the image frame computed for the unknown vehicle. Table 1 shows inter-class and intra-class variation when a van is hypothesized, and for images obtained at four different times in the day. Table 2 shows inter-class and intra-class variation when the car is hypothesized. Such investigation showed that both sets of points A-1 and A-2 produced adequate inter-class separation.

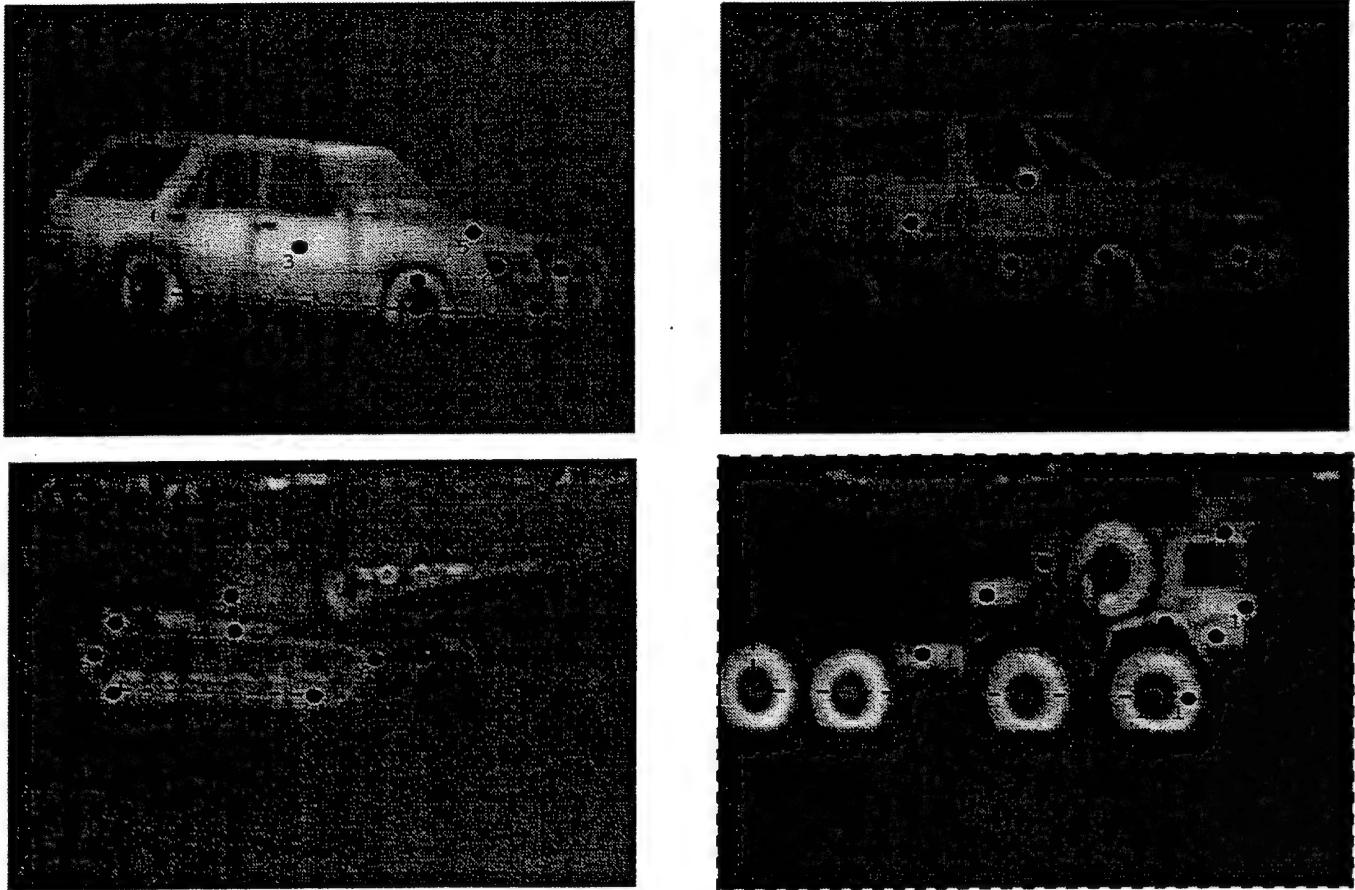


Figure 5: The vehicles used to test the object recognition approach. Clockwise from top left: car, van, truck, and tank. The axes superimposed on the image show the object centered reference frames. The numbered points indicate the object surfaces used to form the measurement matrices. These points are selected such that there are a variety of different materials and/or surface normals within the set. Each vehicle was imaged at different times of the day.

4.2 Sensitivity analysis using simulated data

In order to investigate the robustness of the thermophysical features derived above, the sensitivity of the features to object and scene parameter variations were evaluated. Examination of feature variation was based on simulating the various scene and object energy components for varying scene parameter values - and predicting the feature value. The reader is referred to [Hoekstra, Nandhakumar 1995] for details of the simulation. These simulations were performed on thermophysical object models that were based on equivalent thermal circuits. A widely available circuit simulator was used to implement these circuits. The model was then used to investigate the intra-class invariance and inter-class separability of the features with respect to the following parameters: (1) time of day, (2) day of year, (3) wind speed, (4) object pose, (5) surface absorptivity, and (6) assumed and actual surface emissivity. This approach eliminates the expensive

Time of Day	Hypothesis: Van Data from: Van	Hypothesis: Van Data from: Car	Hypothesis: Van Data from: Truck	Hypothesis: Van Data from: Tank
9 am	13.7716	8.6475	4.4318	10.5850
10 am	13.2269	10.7857	3.7986	15.0399
11 am	13.0597	11.0486	5.2089	11.2887
12 n	12.8853	10.1151	2.8861	22.1023

Table 1: Inter-class variation vs. intra-class variation for feature A-2, consisting of $\{2, 3, 4, 6, 7\}$, and $\{1, 4, 5, 6, 7\}$. Thermophysical properties are chosen for the van hypothesis. The first column shows feature values computed when the correct hypothesis is made and the measurement tensor is obtained from the van. The remaining columns show the results when an incorrect hypothesis is made, i.e. data are collected from the respective objects. This point set exhibits adequate inter-class separation.

Time of Day	Hypothesis: Car Data from: Car	Hypothesis: Car Data from: Van	Hypothesis: Car Data from: Truck	Hypothesis: Car Data from: Tank
9 am	0.4794	28.4142	2.0739	-3.9863
10 am	0.5464	-4.2408	4.8332	-1.7688
11 am	0.4286	0.9313	1.0922	-0.8577
12 n	0.4017	3.8774	0.6181	-0.7631

Table 2: Inter-class variation vs. intra-class variation for feature A-1, consisting of point sets $\{1, 2, 4, 6, 7\}$, and $\{1, 2, 4, 5, 7\}$. Thermophysical properties are chosen for the car hypothesis. The first column shows feature values computed when the correct hypothesis is made and the measurement tensor is obtained from the car. The remaining columns show the results when an incorrect hypothesis is made, i.e. data are collected from the respective objects. This point set exhibits adequate inter-class separation.

and impractical task of collecting real imagery under all possible scene conditions. The values of the features themselves closely corresponded to values calculated from real data when available. An example of the behavior of the features during parameter variation is shown in figure 6. In this case the wind speed has been varied from 2 to 20 MPH. Note that feature B contains singularities. The singularities occur when the temperature vs time curves of different points in the scene cross each other. These singularities are highly localized in time, and the two features continue to be well separated. Similar results have been obtained with respect to the variation of the other parameters for each of the features discussed above.

4.3 Site Change Detection Using Ground Truth Data

The usefulness of these thermophysical invariant features for site change detection tasks was examined, experimentally, by analyzing “ground truth” data gathered from a scene as well as a temporal sequence of LWIR imagery from a scene. The first data set consisted of temperature measurements acquired from thermocouples implanted in various types of materials placed in an outdoor scene. The data were collected over a period of five days in mid-November. The collection includes varying weather conditions and has extensive records of the atmospheric pressure, ambient temperature, lighting conditions, etc. The measurements were recorded at closely spaced intervals of a few minutes. The materials included sod, clay, gravel, concrete,

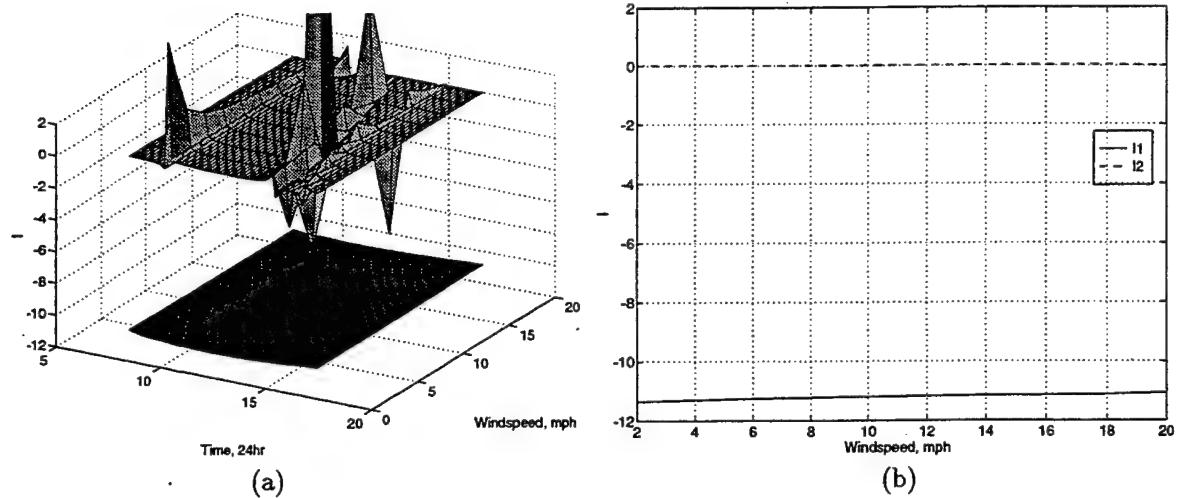


Figure 6: (a) Variation of features B and C with wind speed over the period of an entire day. Feature B consists of points $\{2, 3, 4, 6, 7\}$ and $\{3, 4, 10, 7, 8\}$. Feature C consists of points $\{1, 3, 4, 6, 7\}$ and $\{2, 3, 6, 7, 8\}$. the two distinct features were computed for the Truck vehicle. (b) Variation of features B and C with windspeed at 12 noon.

asphalt, and aluminum. Extensive modeling of the test site has been done, and simulations have reproduced the actual data to within $0.5^{\circ}C$.

Consider a feature constructed (as described in section 3) using point sets {Grass, Clay, Gravel, Asphalt, Aluminum} and {Grass, Clay, Concrete, Asphalt, Aluminum}. The distribution of the value of this feature was computed when the measurements were obtained from the correctly hypothesized materials. The average value of the feature was -91.4, and the standard deviation was 1.7. Next, measurements for one of the hypothesized materials was obtained from a different material – to mimic the situation where one of the materials in the site is changed to a different one. The mean and standard deviation were computed for the feature value under this erroneous hypothesis. For change detection we should notice a significant difference in feature values computed under correct and wrong hypotheses. The feature’s behavior under different wrong hypotheses is summarized in table 3.

Unfortunately, as we increased the number of data points considered, the variance of the thermo-physical invariant increased rapidly. We also noted that the variance did not appear to converge. This is a clear indication that the thermophysical invariants do not fit a Gaussian model. A more appropriate model and a new classification scheme are required, as discussed below.

4.4 Improved Statistical Models for TAI Feature Distributions

Symmetric, Alpha-Stable (S α S) processes form a class of statistical models with several interesting properties, one of which is the lack of finite statistics of order higher than the corresponding characteristic exponent. S α S

Data from	Hypothesized Material					
	Grass	Clay	Gravel	Concrete	Asphalt	Aluminum
Grass	-91.4, 1.7	-91.2, 1.6	-91.1, 2.0	-91.5, 2.3	-90.0, 6.2	-92.0, 5.3
Clay	-72.8, 40.2	-91.4, 1.7	-72.9, 40.2	-91.7, 3.4	-74.4, 42.7	-80.0, 6.1
Gravel	-73.6, 40.8	-70.1, 39.3	-91.4, 1.7	-91.1, 0.1	16.4, 38.8	-84.0, 5.1
Concrete	-89.9, 0.8	-89.7, 0.4	-91.0, 0.1	-91.4, 1.7	-110.4, 0.1	-98.5, 6.2
Asphalt	-56.4, 31.8	-108.9, 90.3	17.8, 50.1	-110.1, 3.9	-91.4, 1.7	-87.9, 7.5
Aluminum	-90.9, 1.7	-91.0, 1.6	-91.2, 1.6	-91.1, 1.6	-91.6, 1.5	-91.4, 1.7

Table 3: The (mean, standard deviation) values for a feature constructed using point sets {Grass, Clay, Gravel, Asphalt, Aluminum} and {Grass, Clay, Concrete, Asphalt, Aluminum}. The values of this feature were computed when the measurements were obtained from the correctly hypothesized materials and also for specific erroneous hypotheses.

processes have been receiving increasing interest from the signal processing and communications communities in the very recent years as statistical-physical models for time series containing severe outliers [Nikias and Shao, 1995]. The performance of Gaussian classifiers on such observations is unacceptably low. However, significant performance gains can be obtained if proper, non-Gaussian classifiers are developed.

A univariate S α S pdf $f_\alpha(\gamma, \delta; \cdot)$ is best defined via the inverse Fourier transform integral [Nikias and Shao, 1995]

$$f_\alpha(\gamma, \delta; x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \exp(i\delta\omega - \gamma|\omega|^\alpha) e^{-i\omega x} d\omega \quad (27)$$

and is completely characterized by the three parameters α (*characteristic exponent*, $0 < \alpha \leq 2$), γ (*dispersion*, $\gamma > 0$), and δ (*location parameter*, $-\infty < \delta < \infty$).

The characteristic exponent α relates directly to the heaviness of the tails of the S α S pdf. The smaller its value, the heavier the tails. The value $\alpha = 2$ corresponds to a Gaussian pdf, while the value $\alpha = 1$ corresponds to a Cauchy pdf. The dispersion γ is a measure of the spread of the pdf, similar to the variance of a Gaussian pdf. Finally, the location parameter, δ , is the point of symmetry of the pdf and equals its mean, whenever the mean is finite (i.e., for $1 < \alpha \leq 2$).

The non-Gaussian ($\alpha \neq 2$) S α S distributions maintain many similarities to the Gaussian distribution, but at the same time differ from it in some significant ways. For example, a non-Gaussian S α S pdf maintains the usual bell shape and, more importantly, non-Gaussian S α S random variables satisfy the linear stability property. However, non-Gaussian S α S pdfs have much sharper peaks and much heavier tails than the Gaussian pdf. As a result, only their moments of order $p < \alpha$ are finite, in contrast with the Gaussian pdf which has finite moments of arbitrary order. These and other similarities and differences between Gaussian and non-Gaussian S α S pdfs and their implications on the design of signal processing algorithms are presented in detail in the tutorial paper [Shao and Nikias, 1992] or in the recent monograph [Nikias and Shao, 1995] to which the interested reader is referred.

The three parameters may be estimated by several methods. The location parameter, δ , may be estimated by the mean for $1 < \alpha \leq 2$, and the median for $\alpha \leq 1$. However, since the data are impulsive in

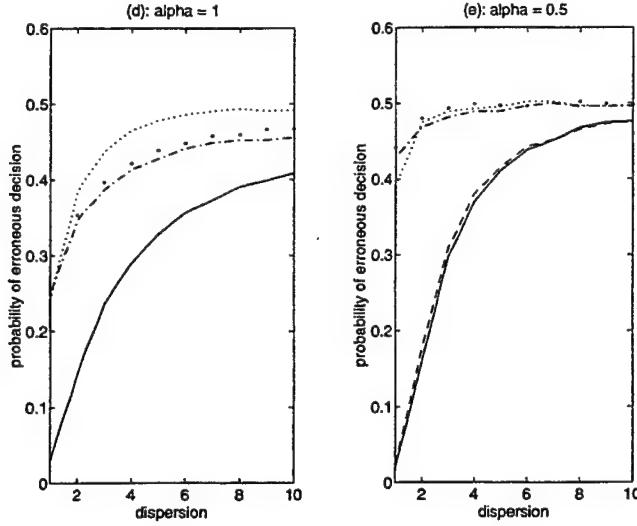


Figure 7: Probability of Error for different classifiers. Solid line: optimal, Point line: linear, Dashed line: Cauchy. The dashed line and the solid line coincide in the plot on the left. The Cauchy classifier performs much better than the Gaussian classifier and only slightly worse than the optimal classifier. The dotted line and dash-dot line are for limiter-integrator schemes with different thresholds.

nature, the median will be more efficient than the mean for all α .

Once the location is known, then the exponent of the tails can be estimated using the $\log |S\alpha S|$ method [Nikias and Shao, 1995].

$$\mathbf{Y} = \text{Var}(\log |\vec{X} - \hat{\delta}|) \quad (28)$$

$$\hat{\alpha} = \sqrt{\frac{1}{\frac{6\mathbf{Y}}{\pi^2} - \frac{1}{2}}} \quad (29)$$

Finally, the dispersion γ can be determined from the data and the estimated δ and α .

$$\hat{\gamma} = \left(\frac{E(|\vec{X} - \hat{\delta}|^p)}{C(p, \hat{\alpha})} \right)^{\frac{\hat{\alpha}}{p}} \quad (30)$$

$$C(p, \hat{\alpha}) = \frac{\Gamma(1 - \frac{p}{\hat{\alpha}})}{\Gamma(1 - p) \cos(\frac{\pi}{2}p)}, \quad (31)$$

where p is the fractional lower order moment used to estimate the dispersion. [Tsihrintzis and Nikias, 1995] show that $p = \alpha/3$ is generally an excellent choice.

The maximum likelihood Cauchy classifier was proposed in [Tsihrintzis and Nikias, 1995] as a robust, yet simple test based on SoS processes. Figure 7 shows that the Cauchy classifier has a minimum probability of error for $\alpha = 1$ and is only slightly sub-optimal for any other α , including $\alpha = 2$! The Gaussian classifier, on the other hand, degrades rapidly for $\alpha < 2$. The Cauchy classifier for a single feature value is:

$$t = \frac{\frac{\gamma_1}{\gamma_1^2 + (\vec{X} - \delta_1)^2}}{\frac{\gamma_2}{\gamma_2^2 + (\vec{X} - \delta_2)^2}} \stackrel{H_0}{>} \stackrel{H_1}{<} \eta \quad (32)$$

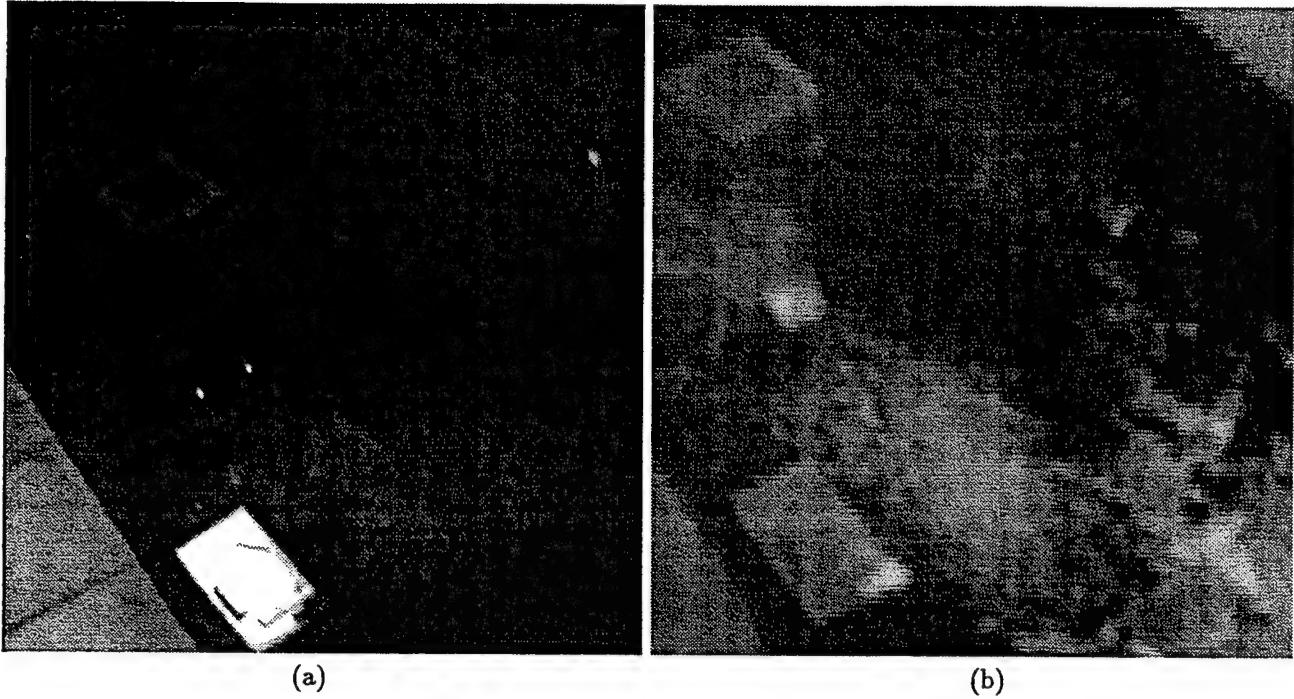


Figure 8: (a) A visual image of the scene used for these experiments. Materials include Asphalt, Concrete, Dirt, Fiberglass, Iron, Slate, and Wood. (b) A corresponding long-wave infrared (LWIR) image. The scene was imaged at 7 different times over a 2 day period.

where η can be varied to change the probability of detection and the probability of false alarm. If more samples are available, then the new test statistic is a sum of the log of the individual statistics. This new statistic can be shown to approach a Gaussian distribution and the probability of error can be reduced on the order of $1/n$, where n is the number of samples available. Having multiple samples available to the original Gaussian classifier will not improve its results.

4.5 Site Change Detection Using Real IR Imagery

In order to test the above ideas on a large collection of real imagery, we first acquired a temporal sequence of IR imagery from an outdoor scene containing asphalt, concrete, dirt, fiberglass, iron, rock, and wood (figure 8). The scene was imaged at 7 different times over a 2 day period. The correct material hypothesis constitutes one class, and a change of one specific material type to another was considered to be a different class. A TAI feature was found that could best detect each change of material. For each (correct as well as wrong) hypothesis, the image data produced 10^9 values of the TAI feature! While the Gaussian model produced estimates of variances on the order of 10^5 – a clear indication of infinite variance [Nikias and Shao, 1995], a good fit to a symmetric alpha-stable model was obtained. Classifiers were constructed to separate

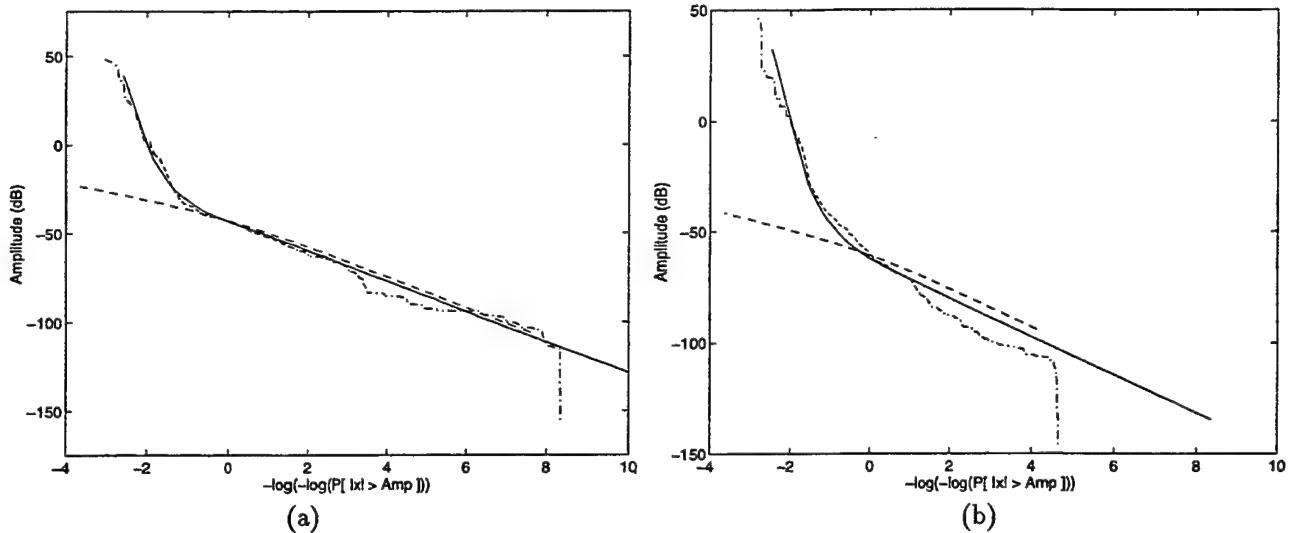


Figure 9: (a) Theoretical SaS Amplitude Probability Distribution (APD) with $\alpha = 1.3$ and $\gamma = 0.001$ ($\delta = 0.1325$) matches the correct hypothesis data very well. The dashed line is the best Gaussian (hand-fitted $\sigma = .008$) approximation. Notice that in the tail region (The upper-left curved part) the Gaussian model diverges from the data. (b) An incorrect hypothesis can also be modeled very well ($\alpha = 0.8$, $\gamma = 0.0075$, and $\delta = 0.0095$, Gaussian $\sigma = .001$).

the correct and erroneous hypotheses.

We discuss, for the purposes of illustration, one specific thermophysical invariant that was chosen to detect a material change from slate to concrete. Figure 9(a) illustrates that the amplitude probability distribution of the feature and the alpha-stable model agree very well. Figure 9(b) shows that the feature distribution under an incorrect hypothesis can also be modeled by a SaS distribution. It is reasonable to expect the TAI feature will be non-Gaussian because the ratio of determinants is essentially a ratio of Gaussian distributions which results in a Cauchy distribution. This clearly indicates that classifiers based on SaS processes will perform better than conventional classifiers designed under the Gaussian assumption.

Using the Cauchy classifier on the data that were modeled in Figure 9, one can achieve a $\Pr[\text{error}] = 2\%$. The Gaussian classifier, on the other hand, predicts a $\Pr[\text{error}] = 49.9\%$! Therefore the Gaussian classifier will not be able to detect if concrete had been poured over slate, but the Cauchy classifier can do so with a low probability of missed detection of change.

5 Discussion

The approach described above is promising in that it makes available features that are (1) invariant to scene conditions, (2) able to separate different classes of objects, and (3) based on physics based models of the many phenomena that affect LWIR image generation. We have also described a scheme for the construction

of robust classifiers based on appropriate models for the distributions of the TAI features.

The specification of optimal sets of points for high inter-class separation and low intra-class variation is a crucial task in this approach. This is a complex search problem, and it is not clear that a solution will always exist for a collection of object classes. Note that different aspects of an object may be imaged – the set of visible points differ for each aspect. The complexity of the search task is compounded by attempting to ensure inter-class separation in the presence of erroneous pose hypothesis.

Some criteria for the choice of point sets are obvious. Points should not be chosen such that the measurement matrix has less than full rank. This occurs, for example, when the five points lie on surfaces with identical surface normals and also have identical thermophysical properties. When the two sets of points have four points in common, and when the remaining two points are on different parts of the imaged object, but lie on materials that are identical/similar, then these point sets will form an invariant that has magnitude of one. Such invariants are useful only if there exists a unique set, or limited number of sets, of points that produces this value. Other thermophysical criteria for point set selection need to be investigated.

The hypothesis of object pose and identity is best achieved by employing geometric invariance techniques [Forsyth et al., 1991]. For example, conics may be fit to wheels which manifest high contrast in LWIR imagery, and their parameter values may be used to compute GI's. This may be employed to generate object identity and pose that may be verified by the thermophysical invariance scheme described above. Future effort will be devoted to: the integration of the above scheme with GI's to produce a complete system, the study of the nature of scene-to-scene transformation of driving conditions, and a detailed exploration of the performance of the scheme when applied to a significant collection of objects, aspects, and scene conditions.

References

- [1] E. Barrett, G. Gheen, and P. Payton, "Algorithms for Invariant Model Transfer and Object Recognition", *Proc ARPA Image Understanding Workshop*, Monterey, CA, Nov 13-16, 1994, pp. 1429-1442.
- [2] T. Binford, T.S. Levitt, and W.B. Mann, "Bayesian Inference in Model-Based Vision", *Uncertainty in AI*, 3, L.N. Kanal, T.S. Levitt, and J.F. Lemmer, (Ed's), Elsevier, 1989.
- [3] J.B. Burns, R.S. Weiss, and E.M. Riseman, "View Variation of Point-Set and Line-Segment Features", *IEEE Trans PAMI*, vol 15, no 1, Jan 1993.
- [4] O. Faugeras and B. Mourrain, "On the geometry and algebra of the point and line correspondences between N images", *Proc. IEEE ICCV*, Cambridge, MA, June 20-23, 1995, pp. 951-956.
- [5] D. Forsyth, J.L. Mundy, A. Zisserman, C. Coelho, A. Heller, C. Rothwell, "Invariant Descriptors for 3D Object Recognition and Pose", *IEEE Trans PAMI*, vol 13, no 12, Oct 1991
- [6] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, San Diego, CA, 1990
- [7] M.J. Gauder, V.J. Velten, L.A. Westerkamp, J. Mundy, and D. Forsyth, "Thermal Invariants for Infrared Target Recognition", *ATR Systems and Technology Conf*, 1993.
- [8] G.B. Gurevich, "Foundations of the Theory of Algebraic Invariants", (translated by J.R.M. Radock and A.J.M. Spencer) P. Noordhoff Ltd - Groningen, The Netherlands, 1964
- [9] G. Healey, "Using Color to Segment Images of 3-D Scenes", *Proc SPIE Conf Applications of AI*, vol. 1468, 1988, Orlando, FL, pp. 814-825.

- [10] G. Healey and D. Slater, "Using Illumination Invariant Color Histogram Descriptors for Recognition", *Proc IEEE Conf CVPR*, June 21-24, 1994, Seattle, WA, pp. 355-360.
- [11] D. Hilbert, *Theory of Algebraic Invariants*, Cambridge University Press, transcribed in 1897, English Translation first published in 1993.
- [12] F.P. Incropera and D.P. DeWitt, *Fundamentals of Heat Transfer*, John Wiley and Sons, New York, 1981.
- [13] G.J. Klinker, S.A. Shafer and T. Kanade, "Image Segmentation and Reflection Analysis through Color", *Proceedings of DARPA Image Understanding Workshop*, Cambridge, MA, 1988, pp 838 - 853.
- [14] D.G. Lowe, "The Viewpoint Consistency Constraint," *International Journal of Computer Vision*, vol. 1, no. 1, 1987, pp. 57-72.
- [15] G. Hoekstra and N. Nandakumar, "Quasi-Invariant Behavior of Thermophysical Features for Interpretation of Radiometric Imagery", *Optical Engineering* to appear 1995.
- [16] J.D. Michel and N. Nandakumar, "Unified Octree-Based Object Models for Multisensor Fusion", *Proc 2nd IEEE Workshop on CAD Model-Based Vision*, Champion, PA, 1994.
- [17] N. Nandakumar and J.K. Aggarwal, "Integrated Analysis of Thermal and Visual Images for Scene Interpretation", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 4, July 1988, pp. 469-481.
- [18] N. Nandakumar and J.K. Aggarwal, "Thermal and Visual Information Fusion for Outdoor Scene Perception", *Proc. of IEEE International Conference on Robotics and Automation*, Philadelphia, PA, April 1988, pp. 1306-1308.
- [19] N. Nandakumar, "A Phenomenological Approach to Multisource Data Integration: Analyzing Infrared and Visible Data", *Proc. IAPR TC7 Workshop on Multisource Data Integration in Remote Sensing*, College Park, MD, June 14-15, 1990.
- [20] N. Nandakumar, "Robust Physics-Based Analysis of Thermal and Visual Imagery", *Journal of Optical Society of America*, JOSA-A, Special Issue on Physics-Based Computer Vision, vol 11, no 11, Nov 1994, pp. 1-9.
- [21] N. Nandakumar, V.J. Velten and J.D. Michel, "Thermophysical Affine Invariants from IR Imagery for Object Recognition", *IEEE Computer Society Workshop on Physics Based Models for Computer Vision*, Cambridge, MA, June 18-20, 1995, pp. 48-54.
- [22] S.K. Nayar and R.D. Bolle, "Reflectance Ratio: A Photometric Invariant for Object Recognition", *Proc IEEE ICCV*, 1993.
- [23] C.L. Nikias and M. Shao, *Signal Processing with Alpha-Stable Distributions*, John Wiley and Sons, New York, 1995.
- [24] T.H. Reiss, "Recognizing Planar Objects Using Invariant Image Features", *Lecture Notes in Computer Science*, 676, G. Goos and J. Hartmanis (Eds), Springer-Verlag, Berlin, 1993.
- [25] E. Rivlin and I. Weiss, "Semi-Local Invariants", *Proc IEEE CVPR* 1993, pp. 697-698
- [26] A. Shashua and M. Werman, "Trilinearity of three perspective views and its associated tensor", *Proc. IEEE ICCV*, Cambridge, MA, June 20-23, 1995, pp. 920-925.
- [27] M. Shao and C.L. Nikias, "Signal Processing with fractional lower-order moments: Stable processes and their applications", *Proc IEEE*, vol 81, pp 986-1010, 1993
- [28] G.A. Tsihrintzis and C.L. Nikias, "Performance of Optimum and Suboptimum Receivers in the Presence of Impulsive Noise Modeled as an Alpha-Stable Process", *IEEE Transactions on Communications*, vol. COM-43, pp. 904-914, 1995
- [29] D. Weinshall, "Direct Computation of Qualitative 3D Shape and Motion Invariants", *IEEE Trans PAMI*, vol 13, no 12, Dec 1991.
- [30] D. Weinshall, "Model-based Invariants for 3D Vision", *Proc IEEE CVPR* 1993, pp. 695-696
- [31] I. Weiss, "Noise-Resistant Invariants of Curves", *IEEE Trans PAMI*, vol 15, no 9, July 1993, pp. 943-948
- [32] L.B. Wolff, "Polarization-based material classification from specular reflection", *IEEE Trans PAMI*, Nov 1990, pp 1059-1071.
- [33] M. Zerroug and R. Nevatia, "Quasi-Invariant Properties and 3D Shape Recovery of Non-Straight, Non-Constant Generalized Cylinders", *Proc IEEE CVPR* 1993, pp 96-103
- [34] R. Zhang, P.-S. Tsai, J.E. Cryer, M. Shah, "Analysis of Shape from Shading Techniques", *Proc IEEE CVPR*, Seattle, WA, 1994, pp. 377-384.

Angle Estimation in the Presence of a Near Field Scatter

Krishna M. Pasala
Professor
Department of Electrical Engineering

University of Dayton
Dayton, OH 45469

Final Report for:
Summer Research Extension Program
Wright Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, Washington, D.C.

and

Wright Laboratory

December 1995

Abstract

The problem of angle estimation using MUSIC algorithm in the presence of a near field scatterer (NFS) is considered here. The effects of mutual coupling are included as well. It is demonstrated that the NFS adversely affects the resolution capability of the MUSIC algorithm. An electromagnetic model of the array and NFS is developed using a hybrid technique that combines the method of moments (MoM) and the Uniform Theory of Diffraction (UTD). This model is used to generate the "actual" currents and terminal voltages on the array and also serves as a basis to develop a technique to compensate for the mutual coupling. A modified array configuration is devised to suppress the field at the array from the NFS. This modification requires a corresponding modification of the MUSIC algorithm. A number of examples are used to demonstrate the effectiveness of the method in restoring the capabilities of the MUSIC algorithm.

1.0 INTRODUCTION

It is often necessary to estimate the angles of arrival of incident signals. It is desirable to accomplish this with a high resolution direction finding algorithm such as the MUltiple SIgnal Classification (MUSIC) algorithm since, often, only a small number of channels are available. This algorithm, along with other parametric “super-resolution” algorithms is very susceptible to errors in the signal model [1]. In particular, it has been shown that ignoring the mutual coupling effects between the antenna elements prevents the MUSIC algorithm from resolving two signals with a small angular separation [2, 3, 4]. The resolution of the MUSIC algorithm can be restored by first pre-processing the signal with the terminal impedance matrix derived from the method of moments model of the antenna array [2, 3, 4]. The signals at the antenna array may also be corrupted by an object in the far field of the array [5 - 12]. In this case the multi-path coherent interference is in the form of plane waves. If the object is in the near field of the antenna array, it will scatterer spherical waves and affect the resolution of the incident signals as shown in figure 1. This object may be a nearby airplane wing or the inside of the antenna array’s radome as shown in figure 2. This paper examines a technique for compensating for the effects of a near field scatterer whose location is known.

The effect of a near field scatterer on Space-Time Adaptive Processing (STAP) radar has been investigated [13, 14]. In [13] it is shown that a near field scatterer causes the clutter to spread in the azimuth-Doppler space. It was also shown that the STAP processor produces weights that form a null at the point of the scatterer. Therefore, increasing the number of spatial degrees of freedom is more effective at suppressing the near field scatterer than increasing the number of temporal degrees of freedom. Finally, it is shown that forcing the STAP processor to place a null at the near field scatterer is always a sub-optimum solution compared to allowing the

STAP processor to null the near field scatterer. However, it requires fewer degrees of freedom and approaches the optimum solution as the number of array elements increases.

In this paper the effect of a near field scatterer on a non-adaptive array with only spatial degrees of freedom is considered. In particular, the antenna array configuration is modified to place a null at the point of the scatterer. This array modification can then be incorporated into the signal model and modified MUSIC algorithm. The result is that the resolution of the MUSIC algorithm is restored to that of the ideal case with no scatterer present. This technique, however, requires knowing the location of the scatterer. The new array configuration also decreases the number of effective antenna elements by one. This implies that the number of signals that can be estimated with the new configuration also decreases by one.

This paper is organized as follows: Section 2 demonstrates the deleterious effect of the interference from a near field scatterer on the resolution capability of the MUSIC algorithm. In section 3 an electromagnetic model of the array and the near field scatterer is developed using a hybrid method combining the method of moments and the Uniform Theory of Diffraction (UTD). A technique to null the fields from the near field scatterer and also compensate for the mutual coupling effects is described in section 4. In section 5, a variety of numerical examples are discussed with a view to assess the effectiveness of the method and the relative importance of direct mutual coupling between the elements, mutual coupling through the near field scatterer and the fields scattered by the near field scatterer. Section 6 summarizes the paper.

2.0 EFFECTS OF A NEAR FIELD SCATTERER

The effects of a near field scatterer on the MUSIC algorithm are illustrated using the array and sources in figure 3. In this case, interference in the form of synthetic spherical waves is

incident upon a nine element linear array designed to resolve narrowband signals over an octave bandwidth from 200 MHz to 400 MHz. The array is composed of nine dipoles which are a half a wavelength long at the center of the bandwidth. The dipoles are separated by 0.375 wavelengths. This is the largest spacing that does not produce any grating lobes anywhere within the bandwidth of the array. Grating lobes are undesirable since they yield ambiguous angles of arrival. Each of the array elements is terminated with the complex conjugate of the input impedance. The array is illuminated by a plane wave with an electric field vector parallel to the dipole. In figure 3 only the spherical waves are shown. The two plane waves are at 300 MHz and 305 MHz with angles of incidence at 45° and 55° relative to the x-axis. The spherical waves represent interference from a near field scatterer. These spherical waves are also at 300 MHz and 305 MHz since they simulate the scattering of the incident plane waves.

Figure 4 shows the MUSIC spectrum corresponding to the problem in figure 3 for several different signal to interference ratios (S/I) with the signal to noise ratio (SNR) set at 10 dB. The spectrum is a normalized average of twenty simulations. The ideal spectrum which shows the result of the MUSIC algorithm when no spherical waves are present is also included in figure 4. Notice that as the S/I ratio decreases, the two peaks begin to merge. Notice also that when the S/I ratio is 15 dB the two signals can no longer be resolved. Figure 4 shows that even when the power in the plane waves is much greater than that in the interference, the resolution capability of the MUSIC algorithm is greatly affected.

3.0 ELECTROMAGNETIC MODEL OF THE ARRAY AND SCATTERER

The array and the scatterer are modeled using a hybrid technique that combines the method of moments (MoM) and the Uniform Theory of Diffraction (UTD) [18, 19]. The array is

modeled using the MoM and the scatterer is with the scatterer and coupling through it are modeled using the UTD. Piecewise sinusoidal modes are used for both the weighting and testing functions in the method of moments to produce an accurate solution with a relatively small number equations [18]. The piecewise sinusoidal expansion of the current on a dipole is shown in figure 5. In this case the current shown is composed of three piecewise sinusoids. The system of equations is solved to yield the current at all points on the antenna array.

The system of equations which incorporates the scatterer is given by [2, 18, 19].

$$\sum_{n=1}^N (Z_{mn} + Z_{mn}^g) I_n = (V_m + V_m^g) \quad m = 1, 2, \dots, N, \quad (1)$$

where N is the number of piecewise sinusoidal modes, Z_{mn} is the mutual coupling between the m^{th} and n^{th} modes, Z_{mn}^g is the mutual coupling between the m^{th} and n^{th} modes through the scatterer, I_n is the current at the n^{th} mode, V_m is the excitation at the m^{th} mode due to the incident electric field, and V_m^g is the excitation at the m^{th} mode due to the incident field scattered by the nearby object. The terms Z_{mn} , Z_{mn}^g , V_m , and V_m^g are calculated using the geometry of the array and scatterer as described in [2]. The current at all points on the antenna array is then found by solving equation (1). This model is used to compute the “actual” currents and voltages at the terminals of the array. The elements of the impedance matrix computed here are also useful in obtaining the terminal impedance matrix described below.

4.0 COMPENSATION FOR A NEAR FIELD SCATTERER

To suppress the effects of a near field scatterer, the excitation at the antenna elements due to the NFS must be eliminated. In addition, the mutual coupling between the antenna elements must also be compensated for. It is possible to obtain, from the actual currents at the antenna

terminals, the terminal voltages that would exist if there were to be no mutual coupling between the elements of the array [3]. Let

$$\vec{I}_a = [I_{1a} \ I_{2a} \ \cdots \ I_{Ma}]^T \quad (2)$$

be the values of the actual terminal currents. Then the corrected terminal voltage vector \bar{V}_c is given by,

$$\bar{V}_c = Z_{kl}^{Te} \vec{I}_a, \quad (3)$$

where Z_{kl}^{Te} is the terminal impedance matrix, whose elements, Z_{kl}^{Te} represent the mutual coupling between the k^{th} and l^{th} terminals. Expressions are derived in [3, 4] that enable the computation of Z_{kl}^{Te} from a knowledge of Z_{mn} , the coupling between the m^{th} and the n^{th} modes. When a scatterer is present, the mutual coupling between antenna elements consists of two terms. The first term is the direct mutual coupling between the antenna elements. The second term is the mutual coupling between the two elements through the scatterer. As shown in [2] all the mutual coupling between the antenna terminals with a scatterer present can be calculated using equation (4)

$$Z_{kl}^{Te} = \frac{1}{I_k^{Te} I_l^{Te}} \sum_{m=1+(k-1)N_m}^{kN_m} \sum_{p=1+(l-1)N_m}^{lN_m} \left[Z_{mp} + \sum_{q=1}^{N_s} Z_{mqp} \right] I_m I_p, \quad (4)$$

where Z_{kl}^{Te} is the mutual coupling between antenna terminals k and l , Z_{mqp} is the mutual coupling between modes m and n through the q^{th} segment of the scatterer. I_k^{Te} is the current at terminal k , I_l^{Te} is the current at terminal l , N_m is the number of piecewise sinusoidal modes on one antenna element, and N_s is the number of segments comprising the scatterer. In general, since the scatterer is in the near field of the antenna array, it is decomposed into several segments so that each segment is in the far field of all the current modes on the antenna array. The terminal

impedance matrix computed using equation (4) may be used to compensate for the mutual coupling effects by computing the corrected voltages using equation (3).

It is still necessary to eliminate the interference due to the near field scatterer. In the next section, a technique is described which effectively puts a null at the scatterer leaving the excitation at the antenna terminals to be entirely from the incident plane waves.

4.1 Near Field Scatterer Nulling Technique

The technique used to suppress the interference from the scatterer is based on a modified antenna array configuration. This modification yields a new signal model and leads to a modified MUSIC algorithm. The antenna elements are combined in pairs as shown in figure 6. The weights are chosen such that each pair of elements forms a null at the point of the near field scatterer as shown in figure 7. It is shown in [2] that the weights should be chosen to satisfy the equation

$$w_i = -\frac{r_{i+1}}{r_i} e^{-j\beta(r_i - r_{i+1})}, \quad (5)$$

where w_i is the weight at the i^{th} antenna pair, r_i is the distance from the i^{th} antenna element to the scatterer, and β is the propagation constant. This equation is derived based upon the assumption that the amplitude of the field scattered by the NFS remains relatively constant over adjacent antenna elements. Notice that each pair of elements which becomes like a new antenna element shares one of its original elements with another pair of elements. Therefore, the noise output at the new elements is correlated. That is, the noise covariance matrix corresponding to the near elements is not a diagonal matrix. Next, it is necessary to determine the effects of combining the elements in pairs on the MUSIC algorithm. This is accomplished by first developing a model for the signal at the outputs of the new antenna elements.

4.2 Signal Model

The signal at the output of the i^{th} new antenna element in figure 6 is given by the expression,

$$x_i(t) = \sum_{k=1}^K u_k(t) [e^{+j\beta d_i \cos(\phi_k)} + w_i e^{+j\beta d_{i+1} \cos(\phi_k)}] + n_i(t) + w_i n_{i+1}(t), \quad i = 1, 2, \dots, M-1, \quad (6)$$

where K is the number of independent far field sources, $u_k(t)$ is the k^{th} narrowband plane wave, β is the propagation constant, d_i is the distance from the origin to the i^{th} antenna element, ϕ_k is the incident angle of the k^{th} signal, $n_i(t)$ is the noise at the i^{th} antenna element, and w_i is the complex weight for the i^{th} antenna pair. Using matrix notation, equation (6) becomes

$$\bar{x} = A\bar{u} + \bar{n}_a + \bar{n}_b, \quad (7)$$

where,

$$A = \begin{bmatrix} e^{+j\beta d_1 \cos\phi_1} + w_1 e^{+j\beta d_2 \cos\phi_1} & \dots & e^{+j\beta d_1 \cos\phi_K} + w_1 e^{+j\beta d_2 \cos\phi_K} \\ \vdots & \ddots & \vdots \\ e^{+j\beta d_{M-1} \cos\phi_1} + w_{M-1} e^{+j\beta d_M \cos\phi_1} & \dots & e^{+j\beta d_{M-1} \cos\phi_K} + w_{M-1} e^{+j\beta d_M \cos\phi_K} \end{bmatrix}, \quad (8)$$

$$\bar{u} = [u_1(t) \ u_2(t) \ \dots \ u_K(t)]^T, \quad (9)$$

$$\bar{n}_a = [n_1(t) \ n_2(t) \ \dots \ n_{M-1}(t)]^T, \quad (10)$$

$$\bar{n}_b = [w_1 n_2(t) \ w_2 n_3(t) \ \dots \ w_{M-1} n_M(t)]^T, \quad (11)$$

$$\bar{x} = [x_1(t) \ x_2(t) \ \dots \ x_{M-1}(t)]^T. \quad (12)$$

The covariance matrix is given by

$$R_x = E[\bar{x} \bar{x}^H] = E[(A\bar{u} + \bar{n}_a + \bar{n}_b)(A\bar{u} + \bar{n}_a + \bar{n}_b)^H], \quad (13)$$

where $E[\bullet]$ denotes mathematical expectation. As shown in [2], equation (13) reduces to

$$R_x = AR_u A^H + \sigma^2 \Sigma_n, \quad (14)$$

where R_u is the covariance matrix of the incident signals given by the expression

$$R_u = E[\vec{u}\vec{u}^H], \quad (15)$$

and $\Sigma_n = \begin{bmatrix} 1 + |w_1|^2 & w_1 & 0 & \dots & 0 \\ w_1^* & 1 + |w_2|^2 & w_2 & \dots & 0 \\ 0 & w_2^* & 1 + |w_3|^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & w_{M-2} \\ 0 & 0 & 0 & w_{M-2}^* & 1 + |w_{M-1}|^2 \end{bmatrix}. \quad (16)$

The matrix Σ_n shows that the noise signals at the output of the new antenna elements are correlated with their neighboring elements. By applying the Mahalanobis transformation [17],

$$\bar{y} = \Sigma_n^{-1/2} \bar{x}, \quad (17)$$

the covariance matrix becomes,

$$R_y = \Sigma_n^{-1/2} A R_u A^H (\Sigma_n^{-1/2})^H + \sigma^2 I, \quad (18)$$

where I is the identity matrix. The noise sources are now uncorrelated. The MUSIC spectrum is calculated by first decomposing the matrix R_y into its eigenvalues and eigenvectors. The noise eigenvectors are then used with the search vector in equation (19) to compute the spectrum using equation (A-1) in Appendix A. The modified search vector is given by

$$a(\phi) = \Sigma_n^{-1/2} \begin{bmatrix} e^{+j\beta d_1 \cos\phi} + w_1 e^{+j\beta d_2 \cos\phi} \\ \vdots \\ e^{+j\beta d_{M-1} \cos\phi} + w_{M-1} e^{+j\beta d_M \cos\phi} \end{bmatrix} \quad (19)$$

5.0 RESULTS

The MUSIC algorithm with the array configuration in figure 6 and the search vector in equation (19) is investigated for three different cases. In the first case no mutual coupling is present between the array elements or the elements and the source of the interfering signals. Therefore, equation (4) is not needed and the ability of the array configuration in figure 6 with the modified

MUSIC algorithm and the search vector in equation (19) alone to suppress the interfering signals can be investigated. In the second case, a point scatterer is used to generate the interfering signals. In this case there exists mutual coupling between the array elements and the scatterer. Therefore it is necessary to use equation (4) that includes mutual coupling effects along with the new array configuration and the modified MUSIC algorithm to suppress effects of the interfering signals. In the third scenario, the point scatterer is replaced by a distributed scatterer. The interference no longer emanates from a single point. In this case equations (4) and (19) are used to suppress the interference where it is assumed that all the interference can be considered as arriving from a single point. This is a valid assumption in some cases but not in all cases. These cases show the usefulness of the new array configuration in suppressing the effects of scattering from a near field object.

All the simulations except those in section 5.1 are performed on the nine element linear dipole array discussed in section 2.0. This array is designed to resolve angles of arrival over one octave from 200 MHz to 400 MHz. The incident signals are plane waves at frequencies in the center of the operating band at 300 MHz and 305 MHz. This separation is sufficient to decorrelate the signals. The spectra shown are the normalized average of 20 simulations computed at 0.1° interval. In all simulations, the SNR is fixed at 10 dB. The simulated data is composed of 300 snapshots.

5.1 Suppression of the Interference

For the two examples presented in this section, the antenna elements are taken to be ideal and isotropic with no mutual coupling between them. The near field scatterer is simulated by a point source radiating spherical waves at the same frequency as the incident plane waves. Thus

mutual coupling effects are completely eliminated and it is possible to study the effectiveness of the new array configuration and the corresponding modification of the MUSIC algorithm.

The antenna element configuration with the search vector as given in equation (19) is used to resolve plane waves when interference from a near field scatterer is present. The spherical waves emanate from point sources located at the given position and with the same frequency as the incident plane waves. Point sources are taken to be uncoupled to the antenna array. Therefore, the usefulness of the element configuration in figure 6 can be investigated without being concerned about the effects of mutual coupling.

The simulations are performed with two plane waves and two point sources both located at the same point as shown in figure 3. Mathematically, the incident signal is,

$$s_n = A_p e^{-j(n-1)\beta_1 d \cos\phi_1} + A_p e^{-j(n-1)\beta_2 d \cos\phi_2} + A_s \frac{e^{-j\beta_1 r_n}}{r_n} + A_s \frac{e^{-j\beta_2 r_n}}{r_n}, \quad (20)$$

where A_p is the amplitude of the plane waves, n is the antenna element index, β_1 and β_2 are the propagation constants for the two signals, d is the separation between elements, ϕ_1 and ϕ_2 are the angles of arrival of the incident plane waves with respect to the axis of the array, A_s is the amplitude of the spherical waves, and r_n is the distance from the n^{th} antenna element to the point sources.

The simulations are performed for various signal to interference (S/I) ratios. In the first case, the plane wave and spherical wave amplitudes are chosen such that the S/I ratio is 5 dB. In the second case, the S/I ratio is 0 dB. These simulations show the usefulness of the antenna element configuration in figure 6 for suppressing spherical waves.

Figure 8 shows the effect of the spherical waves when the S/I ratio is +5 dB. The spectrum labeled "Plane Waves Only" shows the ideal spectrum without the interference. This spectrum is calculated with the conventional MUSIC algorithm and represents the best possible spectrum under

the given conditions. The spectrum labeled "Plane and Spherical Waves" shows the MUSIC spectrum with no nulling of the interference. Notice that the presence of the spherical waves prevents the resolution of the desired signals. The spectrum labeled "Compensated" shows the output of the MUSIC algorithm with the search vector calculated using equations (5), (16), and (19). Notice that the spectrum is almost identical to the ideal case. Figure 8 shows that when the S/I ratio is 5 dB, the resolution of the MUSIC algorithm can be restored with the modified search vector.

Figure 9 shows the effect of the interference when the S/I ratio is 0 dB. In figure 9, the ideal spectrum is repeated for reference. The "Plane and Spherical Waves" spectrum show that the MUSIC algorithm can not resolve the two plane waves. The compensated spectrum shows that with the new configuration, it is possible to restore, almost completely, the resolution capability of the algorithm.

Figures 8, and 9 show the results of using the antenna element configuration in figure 6 to suppress the interference from a near field scatterer. In both cases the interference adversely affects the resolution capability of the MUSIC algorithm. The resolution of the MUSIC algorithm, however, is restored by using the new antenna element configuration and the corresponding modification of the algorithm.

5.2 Point Scatterer

For all the examples in this section, the antenna array is a 9 element dipole array. The scatterer is represented as a point scatterer so that all the energy scattered by the object can be considered to be arriving from a single point. Unlike the point sources in the previous section, however, the point scatterer is coupled to the antenna array. Therefore, it is necessary to use the terminal impedance matrix along with the new antenna array configuration in figure 6 to

compensate for the effects of the near field scatterer. Since the scatterer is small enough to be considered a point, the terminal impedance matrix in equation (4) is calculated with $N_s = 1$. In this section, the new array configuration, search vector and the terminal impedance matrix are used to suppress the interference from a point scatterer.

Four illustrations are presented in this section. The purpose of these illustrations is to assess the relative importance of the mutual coupling and the energy scattered by the NFS and also the relative importance of direct coupling and coupling through the NFS on the MUSIC spectrum. This is accomplished by varying the RCS of the NFS from -5 dBsm to 5 dBsm and observing its effect on the MUSIC algorithm. In each case compensation is corrected for mutual coupling only, from NFS only, and for both. The results are presented in figures 10 through 13. Each of the figures contains the MUSIC spectrum with compensation for mutual coupling alone, compensation for the signals from the NFS, and compensation for both and the ideal spectrum.

The nine element linear dipole array is designed as described in section 2.0. This array can resolve plane waves over a one octave bandwidth from 200 to 400 MHz. The incident signals are uniform plane waves at 300 and 305 MHz with angles of arrival at 45° and 60°. The angle of arrival of 55° used in the last example is replaced with 60° so that in the absence of a scatterer, the MUSIC algorithm can still resolve the signals with the mutual coupling present. These parameters are used for the nine element dipole array to determine the ability of the array configuration in figure 6 and the terminal impedance matrix in suppressing the effects of a point scatterer.

Illustration 5.2-1

Figure 10 shows the results of compensating for the near field scatterer with a specified RCS of -5 dBsm. The ideal curve shows the spectrum without mutual coupling and without the scatterer present. The solid curve shows the result of compensating for the mutual coupling

including the scatterer but not compensating for the interference from the scatterer with the new array configuration. Notice that the solid curve yields a good estimate of the actual spectrum. This is because the signal from the scatterer whose RCS is -5 dBsm is very small. The dashed curve (denoted by “Corrected for V2” in the figure) corresponds to interference suppression but no mutual coupling compensation. In this case, the spherical waves from the scatterer are suppressed with the array configuration in figure 6 but no compensation is made for the mutual coupling effects. In this case the effect on the spectrum is more significant since the effects of mutual coupling between the antenna elements on the spectrum is stronger than the signal from the scatterer. The final spectrum in figure 10 corresponds to correcting for both mutual coupling and the spherical waves from the scatterer using figure 6 and equations (4) and (19). Notice that the new spectrum is very close to the ideal spectrum. Figure 10 shows that compensating for only the mutual coupling produces a reasonably good estimate of the spectrum in this instance but compensating for both the mutual coupling and the interfering signals from the scatterer yields a spectrum almost as good as the ideal case.

Illustration 5.2

In figure 11 the RCS of the point scatterer is increased to 0 dBsm. Notice that the spectra in figure 11 are very similar to those in figure 10. In this case, the solid curve which represents compensating for only mutual coupling has a spectrum that is worse than that of the corresponding spectrum in figure 10. This occurs since the object whose RCS is larger scatters more energy to the array. The final spectrum shows that compensating for both the mutual coupling and the interference from the scatterer produces a spectrum almost as good as the ideal case.

Illustration 5.2-3

Figure 12 shows the results for a point scatterer with a RCS of 5 dBsm. Notice that the spectrum obtained from correcting for only the mutual coupling is worse than in figures 10 and 11. This is due to the fact that the RCS of the scatterer is the largest of the three cases. Notice that even with the large RCS, the final spectrum yields an estimate of the angles of arrival almost as good as in the ideal case.

Illustration 5.2-4

The effect of mutual coupling is shown to be significant in figures 10 through 12. The mutual coupling between two objects is inversely proportional to the distance between them. Since the antenna elements are separated by 0.375λ the coupling between them is significant. The coupling between the near field object and the antenna array is less significant because the distance between them, in general, is greater. The mutual coupling between the scatterer and the array is less significant than the coupling between the elements themselves. Mathematically, this means that,

$$Z_{mp} \gg \sum_{q=1}^{N_r} Z_{mqp} \quad \text{for every } m \text{ and } p. \quad (21)$$

In other words, equation (4) is approximately the same as,

$$Z_{kl}^{Te} \approx \frac{1}{I_k^{Te} I_l^{Te}} \sum_{m=1+(k-1)N_m}^{kN_m} \sum_{p=1+(l-1)N_m}^{lN_m} Z_{mp} I_m I_p. \quad (22)$$

Therefore, while Z_{mn}^g is still calculated and used to determine the “actual” currents on the array using equation (1), it is not necessary to use it when compensating for the mutual coupling effects. Therefore, it is not necessary to know the location or even the type of scatterer to determine the terminal impedance matrix. Note that in this section the near field object is a point scatterer and therefore $N_s = 1$. The significance of compensating for the mutual coupling between the scatterer and the array is investigated in figure 13. In this case a point scatterer with a RCS of 5

dBsm is placed 0.5λ from the array. The results for this case are shown in figure 14. The ideal spectrum shows the result when no mutual coupling or scatterer is present. The spectrum labeled "actual" shows the result when mutual coupling and the scatterer are present. No compensation is performed for either the mutual coupling or the interference from the scatterer. The spectrum labeled "corrected for antenna coupling" shows the result when the new array configuration and search vector are used but the terminal impedance matrix is calculated with equation (22). This assumes that,

$$\sum_{q=1}^{N_s} Z_{mqp}, \quad (23)$$

is negligible compared to Z_{mp} for all m and p . This spectrum yields a very accurate estimate of the angles of arrival even though a strong scatterer with a RCS of 5 dBsm is near the array. The final spectrum labeled "corrected for all coupling" shows the result of using the new array configuration and equation (4) instead of equation (22). Note that the spectrum is not significantly improved. Figure 14 shows that the coupling through the scatterer may be neglected with only a small loss in the performance of the algorithm.

5.3 Distributed Scatterer

Sections 5.1 and 5.2 show the results when all the interference arrives from a single point. In section 5.1, point sources are used to generate the interference. In this case, the array configuration in figure 6 along with the modified MUSIC algorithm is used to suppress the interference and restore the resolution capability of the MUSIC algorithm. In section 5.2, the point sources are replaced with a point scatterer. In this case, the interference is very similar to the previous case but there is now mutual coupling between the scatterer and the array. This coupling is suppressed with the use of the terminal impedance matrix as given by equation (4). The previous

two sections demonstrate that the interference can be adequately suppressed when it arrives from a single point.

In this section, the scattered energy arrives from a finite size object. However, all the energy can be considered to have one phase center. As shown in figure 15, the scatterer is a 90° wedge with a length of 10 meters. The mutual coupling between the elements and the wedge and the array is suppressed using equation (4). For the wedge scatterer, the energy arriving at the array from the scatterer can be considered as emanating from the endpoints of the wedge which are 10λ apart. However, since the length of the wedge lies in the plane of the electric field vector, the energy can be considered as arriving from the midpoint of the wedge which lies in the xy plane. Therefore, the weights for the array configuration in figure 6 are chosen assuming that the wedge is a point scatterer at the location shown in figure 15.

Figure 16 shows the result of compensating for the distributed scatterer. As usual, the ideal spectrum is included as a reference. The solid curve represents the spectrum after compensation for mutual coupling with equation (4). In this case N_s is much greater than one since the scatterer has significant size. The spectrum labeled "corrected for V2" shows the effect of correcting for the interference from the scatterer but not the mutual coupling between the scatterer and the array or between the array elements themselves. Notice that in this case only one signal is apparent in the spectrum. The final spectrum represents compensating for all the mutual coupling and the interference from the scatterer. Notice that the spectrum is almost as good as the ideal case. Figure 16 shows that the resolution capability of the MUSIC algorithm can be restored for a large scatterer if all the scattered energy can be considered to have one phase center.

6.0 SUMMARY AND CONCLUSIONS

In this paper the effect of a scatterer in the vicinity of an array on the direction of arrival estimation using the MUSIC algorithm is considered. It is shown that the resolution capability of the MUSIC algorithm is adversely affected by the NFS. A technique to suppress the effects of the NFS on the angle of arrival estimation is presented. This technique involves a new array configuration obtained by combining the weighted outputs of pairs of elements. The weights are chosen to null the scattered field from the NFS. The modification in the array configuration alters the signal model. This requires a modification of the MUSIC algorithm. Several numerical examples involving both synthetic and real scatterers are used to demonstrate the effectiveness of the method. It is also shown that the ever present mutual coupling also adversely affects the spectrum and must be compensated for, in addition to suppressing the NFS, to get the best results. The mutual coupling has two components; the direct interelement mutual coupling, and the coupling through the scatterer. In general, for the purpose of compensation, the effect of the coupling through the scatterer is small and may be neglected. The new configuration does reduce the degrees of freedom by one and therefore reduces the number of directions of arrival that may be estimated by one. The technique may be readily extended to the case where there is more than one NFS that must be suppressed by combining properly the outputs of more than two elements, with concurrent reduction in the number of degrees of freedom.

APPENDIX A: MUSIC ALGORITHM

The MUltiple Signal Classification (MUSIC) algorithm [15, 16] is a popular parametric method for estimating the angles of arrival of incident signals. This is primarily because its resolution capability is limited only by the signal to noise ratio. To apply the algorithm, the

covariance matrix at the output of the antenna array is decomposed into its eigenvalues and eigenvectors. The eigenvectors associated with the noise can be separated from the eigenvectors associated with the signal by examining the eigenvalues. The noise eigenvectors are then used to form the noise subspace. The MUSIC spectrum is computed by searching for the plane waves with angles of arrival that are orthogonal to the noise subspace. Mathematically, the spectrum is computed with the equation,

$$P(\phi) = \frac{1}{\sum_{i=K+1}^M \left| \tilde{\beta}_i^H \bar{a}(\phi) \right|^2}, \quad (\text{A-1})$$

where K is the number of independent signals, M is the number of antenna elements, $\tilde{\beta}_i$ is ith noise eigenvector, and $\bar{a}(\phi)$ is the search vector given by the equation

$$\bar{a}(\phi) = [e^{-j\beta d_1 \cos\phi} \quad e^{-j\beta d_2 \cos\phi} \quad \dots \quad e^{-j\beta d_M \cos\phi}]^T. \quad (\text{A-2})$$

Note that if $K \geq M$ then there will be no noise eigenvectors to form the noise subspace and the algorithm can not be used.

7.0 REFERENCES

1. Friedlander, B., Weiss, A. J., "Direction Finding in the Presence of Mutual Coupling" *IEEE Transactions on Antennas and Propagation*, Vol. 39, No. 3, March 1991, pp. 273-284.
2. Friel, E. M., "Direction Finding with Compensation for Electromagnetic Effects", Ph.D. Dissertation, University of Dayton, Dayton, Ohio, December 1995.

3. Friel, E. M., Pasala, K. M., "Wideband Bearing Estimation with Compensation for Mutual Coupling Effects" in *Proceedings of IEEE Antennas and Propagation / URSI Symposium*, Seattle, WA, 1994.
4. Pasala, K. M., Friel, E. M., "Mutual Coupling Effects and their Reduction in Wideband Direction of Arrival Estimation" in *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 30. No. 4. October 1994, pp. 1116-1122.
5. Evans, J. E., Johnson, J. R., Sun, D. F., "Application of Advanced Signal Processing Angle-of-Arrival Estimation in ATC Navigation and Surveillance Systems" MIT Lincoln Lab., Lexington, MA, Rep. 582, 1982.
6. Shan, T. J., Wax, M., Kailath, T., "On Spatial Smoothing for Direction-of-Arrival Estimation of Coherent Signals" *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP - 33, PP. 806-811, 1985.
7. Pillai, S. U., Kwon, B. H., "Forward - Backward Spatial Smoothing Techniques for Coherent Signal Identification" *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP - 37, pp. 8-15, 1989.
8. Williams, R. T., Prasad, S., Mahalanabis, A. K., Sibul, L., "An Improved Spatial Smoothing Technique for Bearing Estimation in Multipath Environment" *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 36. pp. 1361 - 1375, 1988.
9. Du, W., Kirlin, R. L., "Improved Spatial Smoothing for DOA Estimation of Coherent Signals" *IEEE Transactions on Signal Processing*, Vol. 33. pp. 806-811, 1991.

10. Friedlander, B. "Direction Finding with an Interpolated Array", in *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* 1990, Albuquerque, NM, pp. 2951-2954, 1990.
11. Friedlander, B. Weiss, A. J., "Direction Finding Using Spatial Smoothing with Interpolated Arrays" *IEEE Transactions on Aerospace and Electronic Systems* Vol. 28. No. 2, pp. 574-587, April, 1992.
12. Wax, M., Sheinvald, J., "Direction Finding of Coherent Signals via Spatial Smoothing for Uniform Circular Arrays" *IEEE Transactions on Antennas and Propagation*, Vol. 42, No. 5, pp. 613-620., May 1994.
13. Barile, E. C., Fante, R. L., Torres, J. A., "Some Limitations on the Effectiveness of Airborne Adaptive Radar," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 28, No. 4, October 1992.
14. Barile, E. C., Guella, T. P., Lamensdorf, D., "Adaptive Antenna Space-Time Processing Techniques to Suppress Platform Scattered Clutter for Airborne Radar," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 31, No. 1, January 1995.
15. Pillai, S. U., Array Signal Processing New York, Springer-Verlag 1989.
16. Schmidt, R. O., "Multiple Emitter Location and Signal Parameter Estimation" in *Proc. RADC Spectral Estimation Workshop*, Oct. 1979, pp. 243-258; reprinted in *IEEE Transactions on Antennas and Propagation* Vol. AP-34, pp. 276-280, March 1986.
17. Therrien, C. W., Discrete Random Signals and Statistical Signal Processing, Prentice Hall, Englewood Cliffs, NJ
18. Stutzman, W. L., Thiele, G. A., Antenna Theory and Design, New York, Wiley & Sons, 1981.

19. Thiele, G. A., Newhouse, T. H., "A Hybrid Technique for Combining Moment Methods with the Geometric Theory of Diffraction," *Proc. IEEE*, Vol. 62, pp. 1438-1447, 1974.
20. Balanis, C. A., Advanced Engineering Electromagnetics, John Wiley & Sons, 1989.

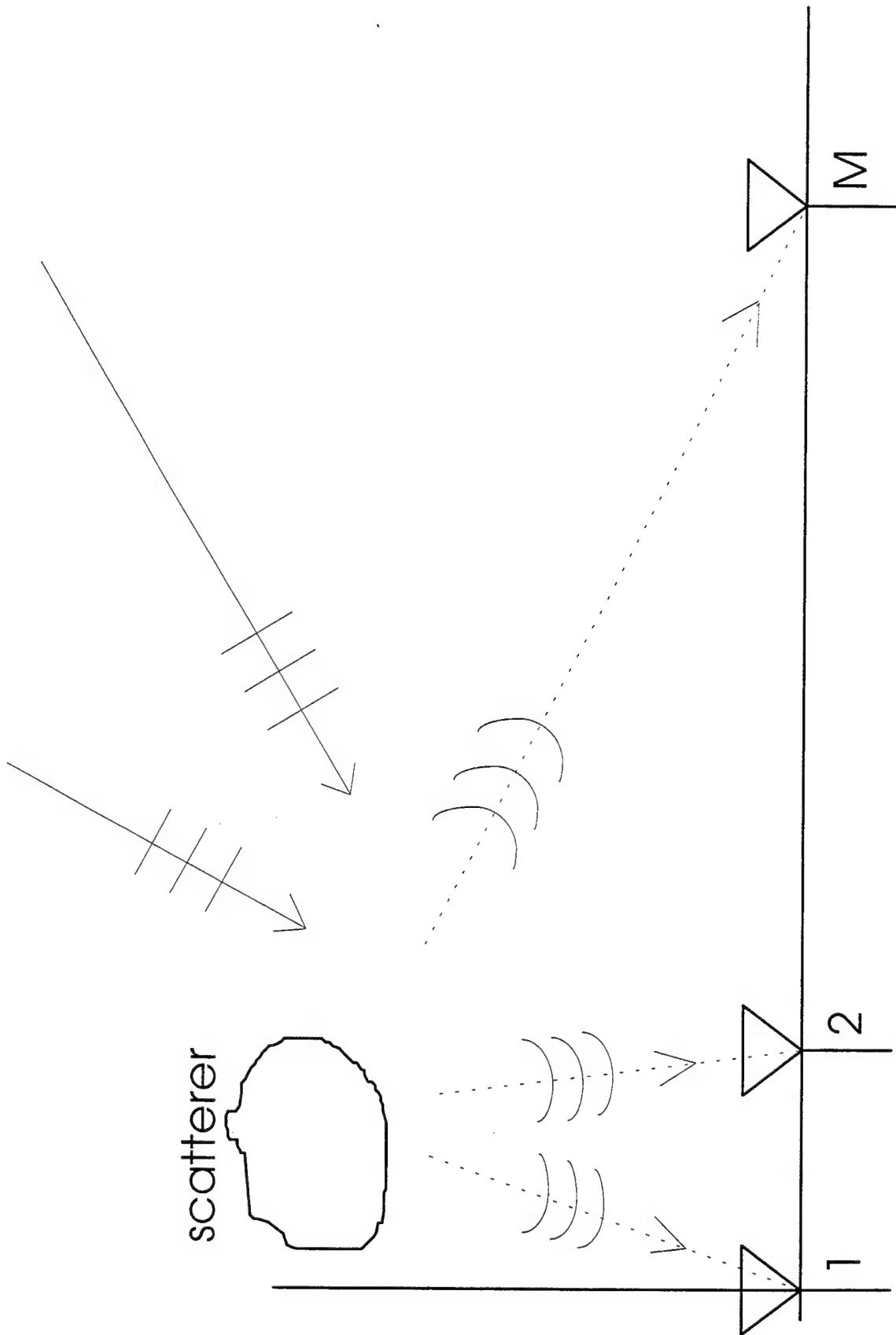


Figure 1: Geometry of Array, Sources, and Scatterer

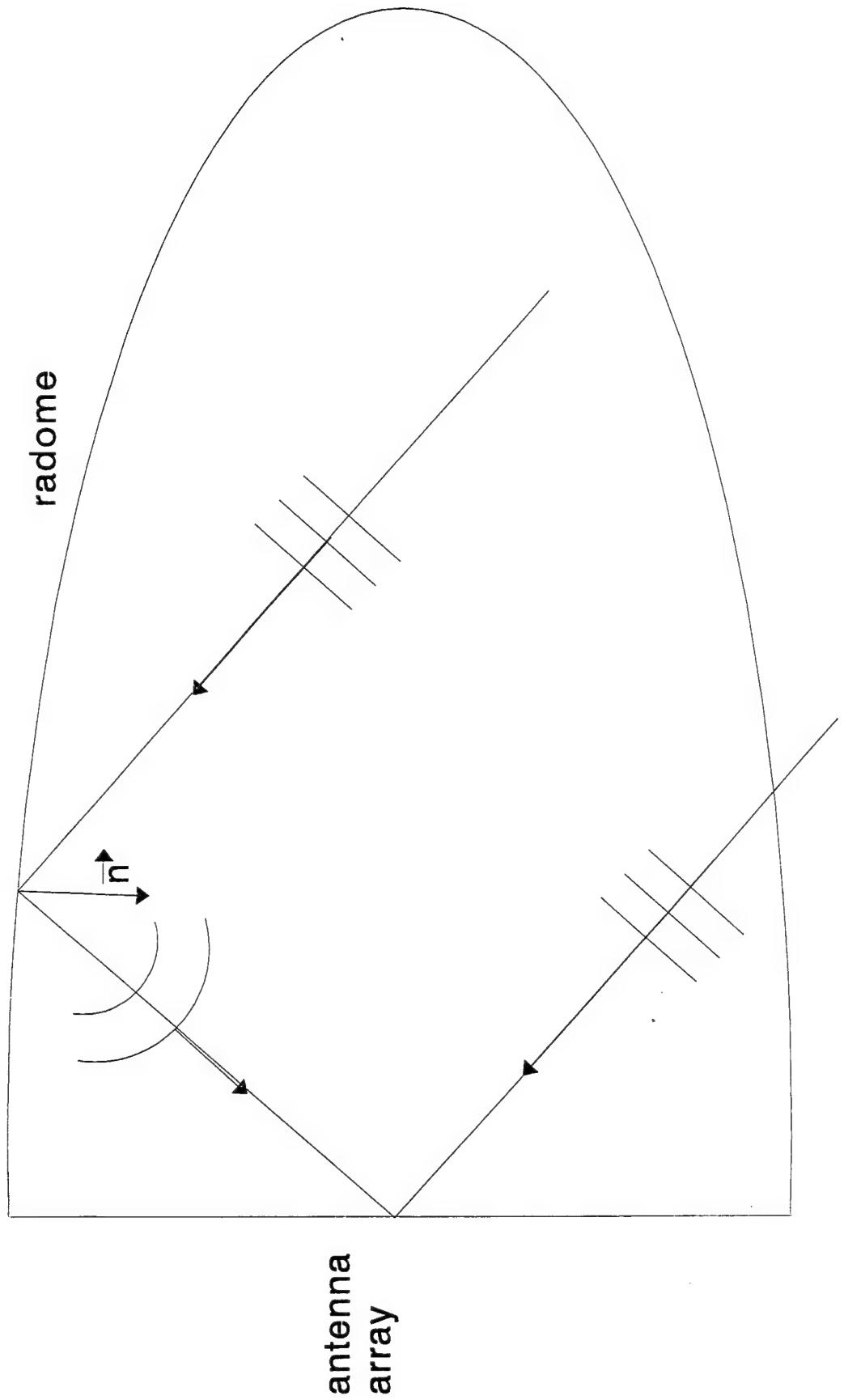


Figure 2: Near Field Scattering by the Inside of a Radome

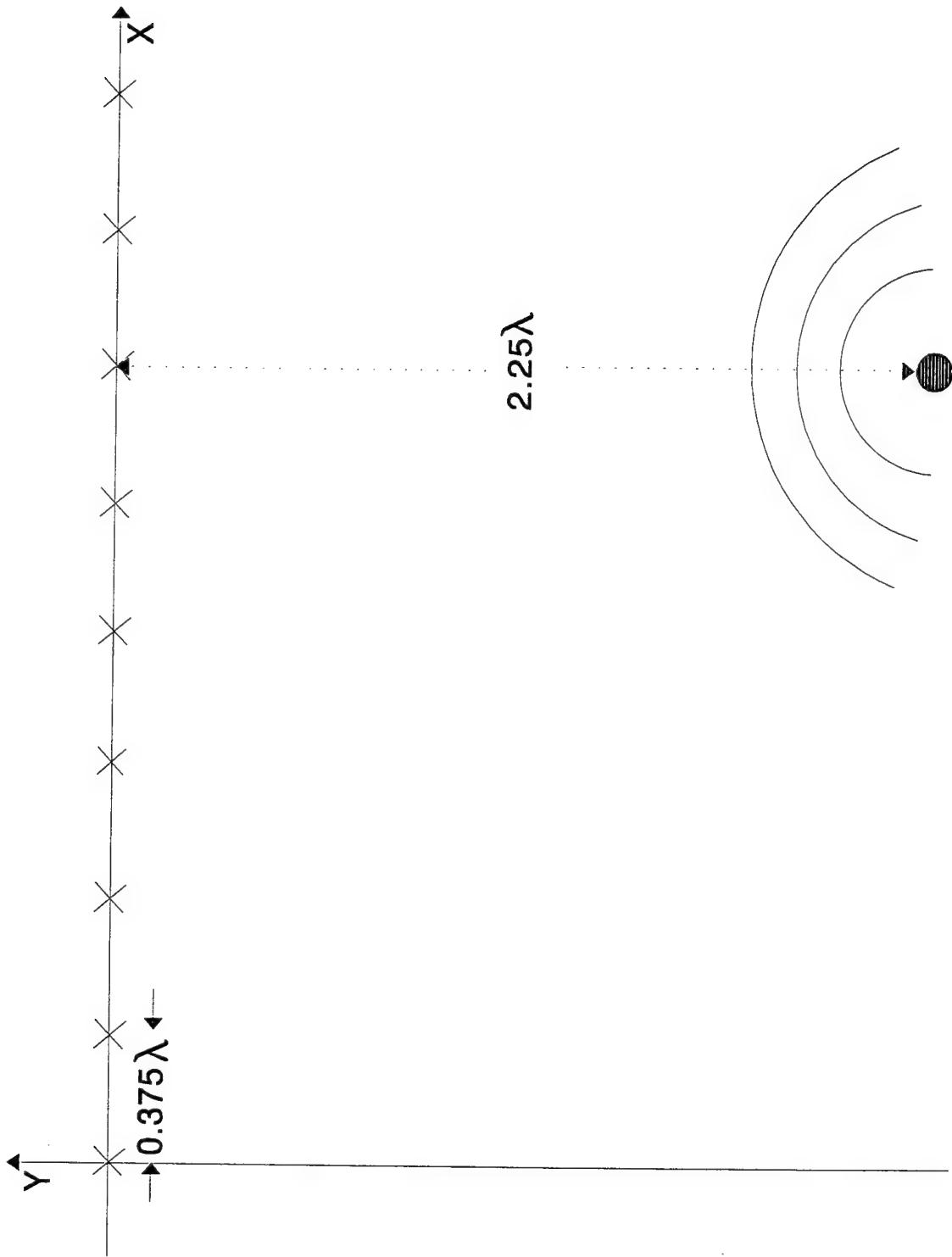


Figure 3: Plane and Spherical Waves Incident Upon an Array

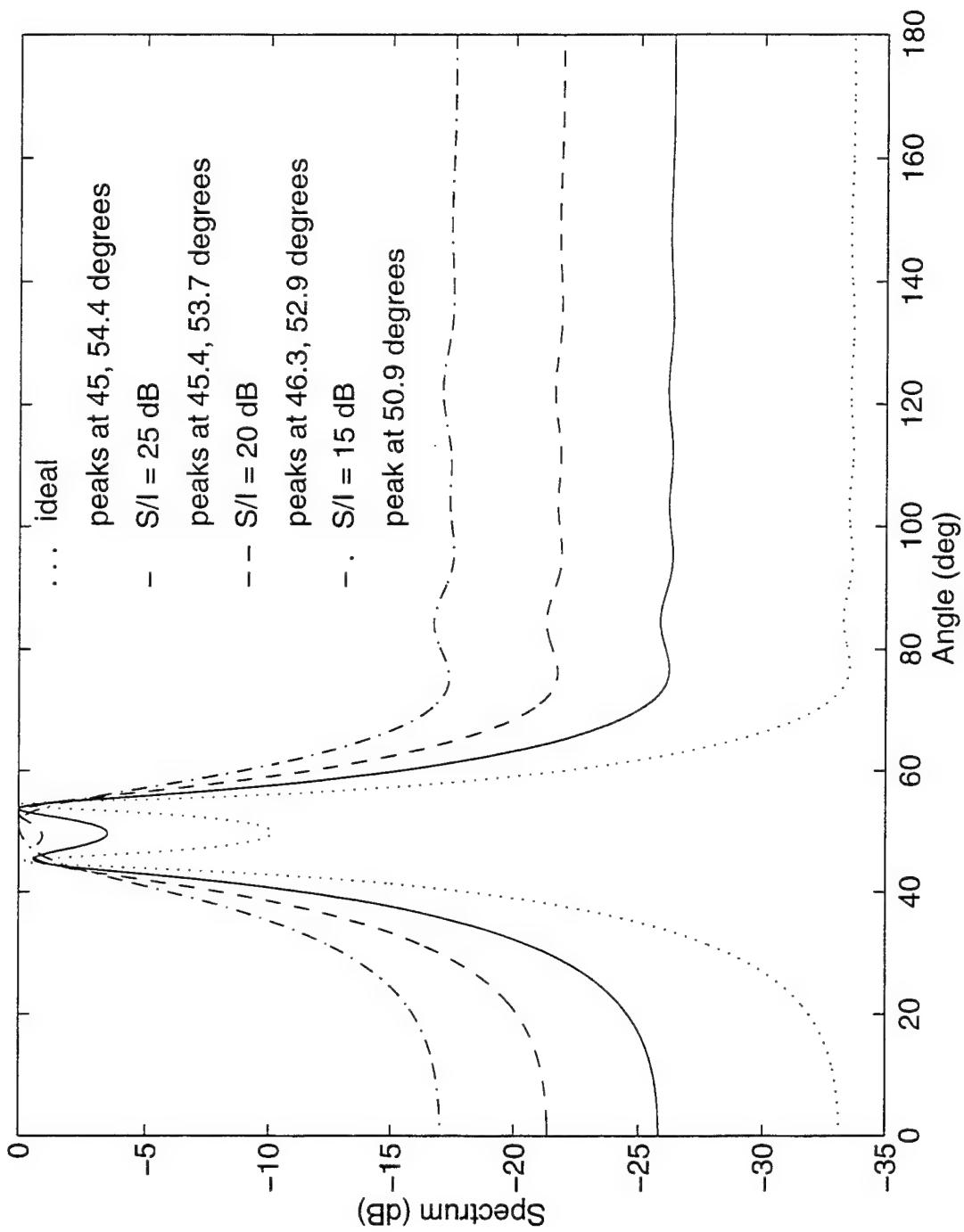


Figure 4: Effect of Spherical Waves on the MUSIC Spectrum

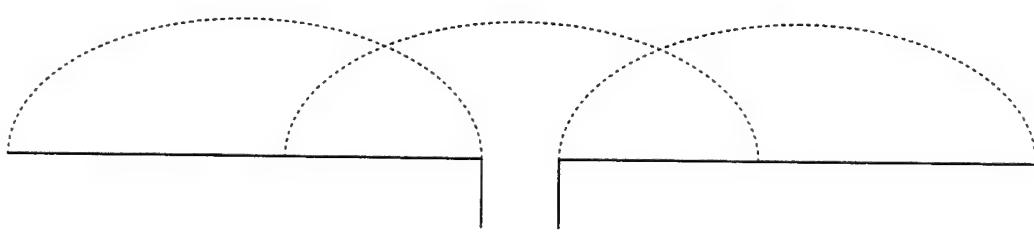


Figure 5: Piecewise Sinusoidal Current Expansion on a Dipole

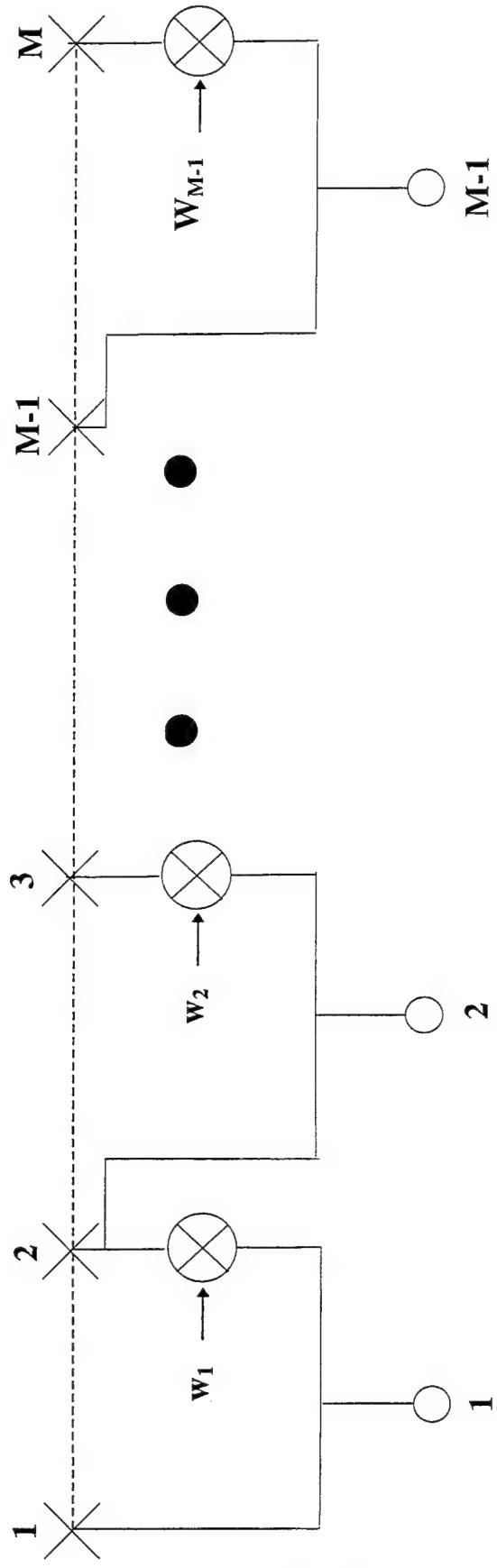


Figure 6: Configuration for Near Field Scatterer Nulling

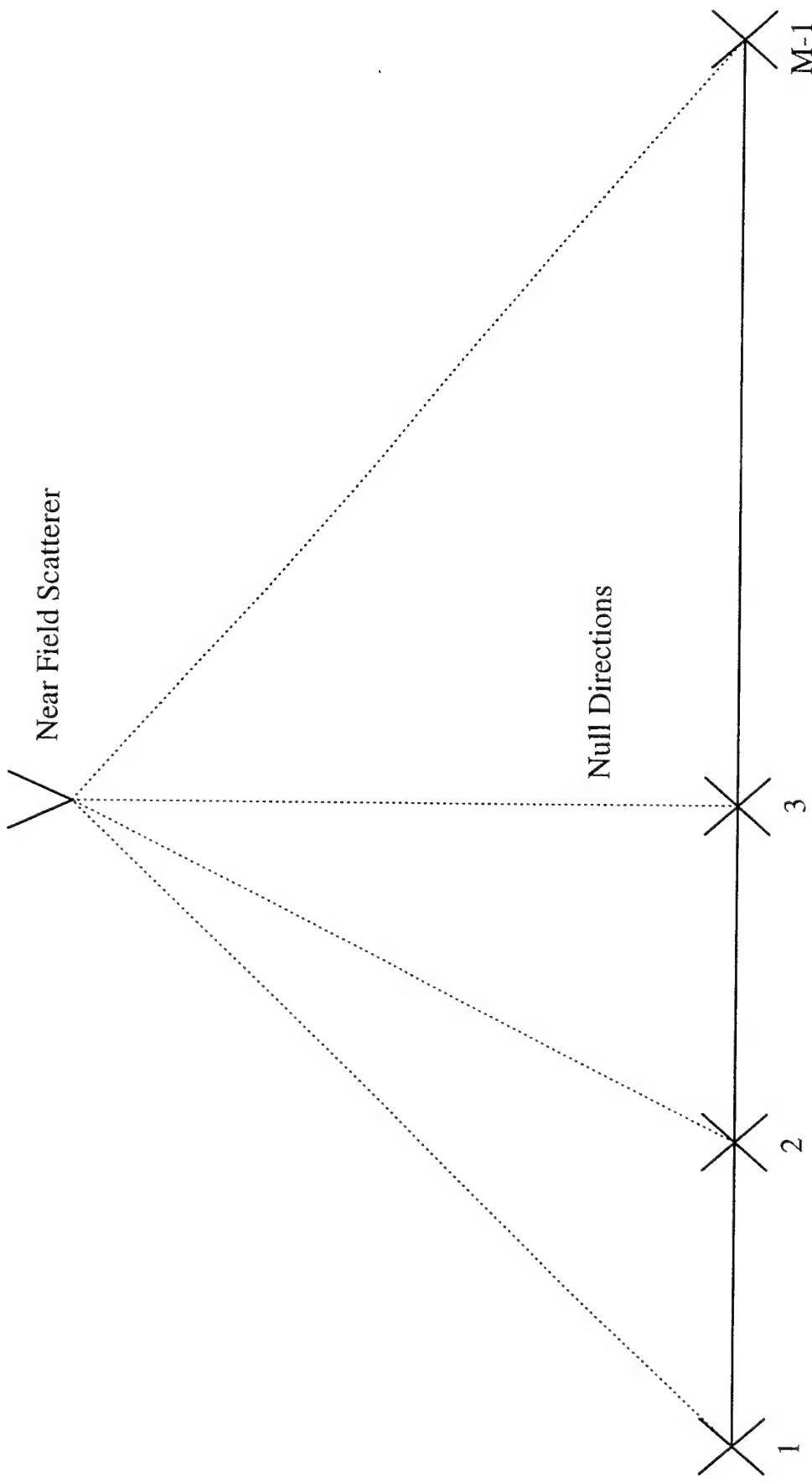


Figure 7: Equivalent Elements with Nulls at the Point of the Scatterer

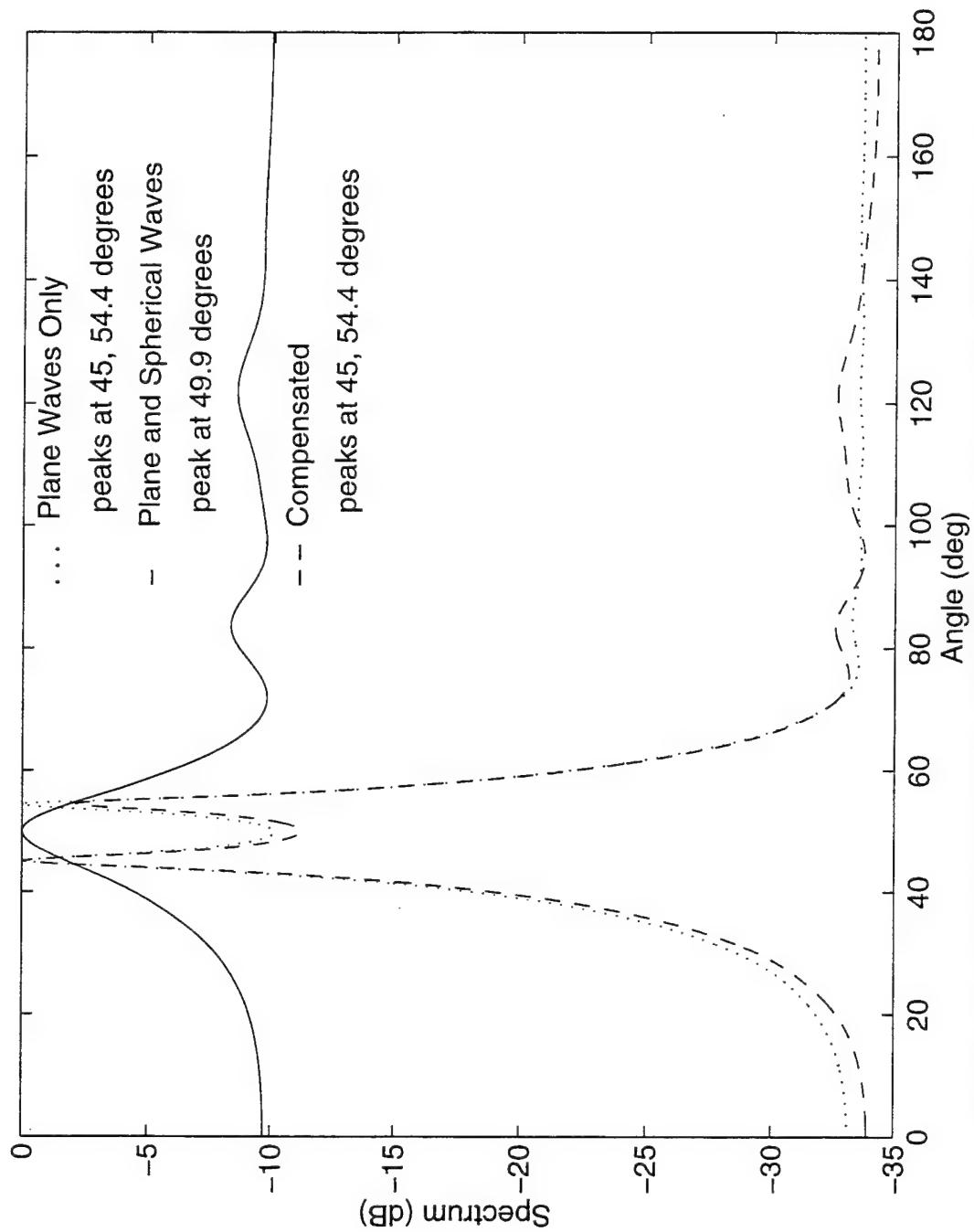


Figure 8: Compensation for Point Sources, $S/I = 5$ dB, SNR = 10 dB

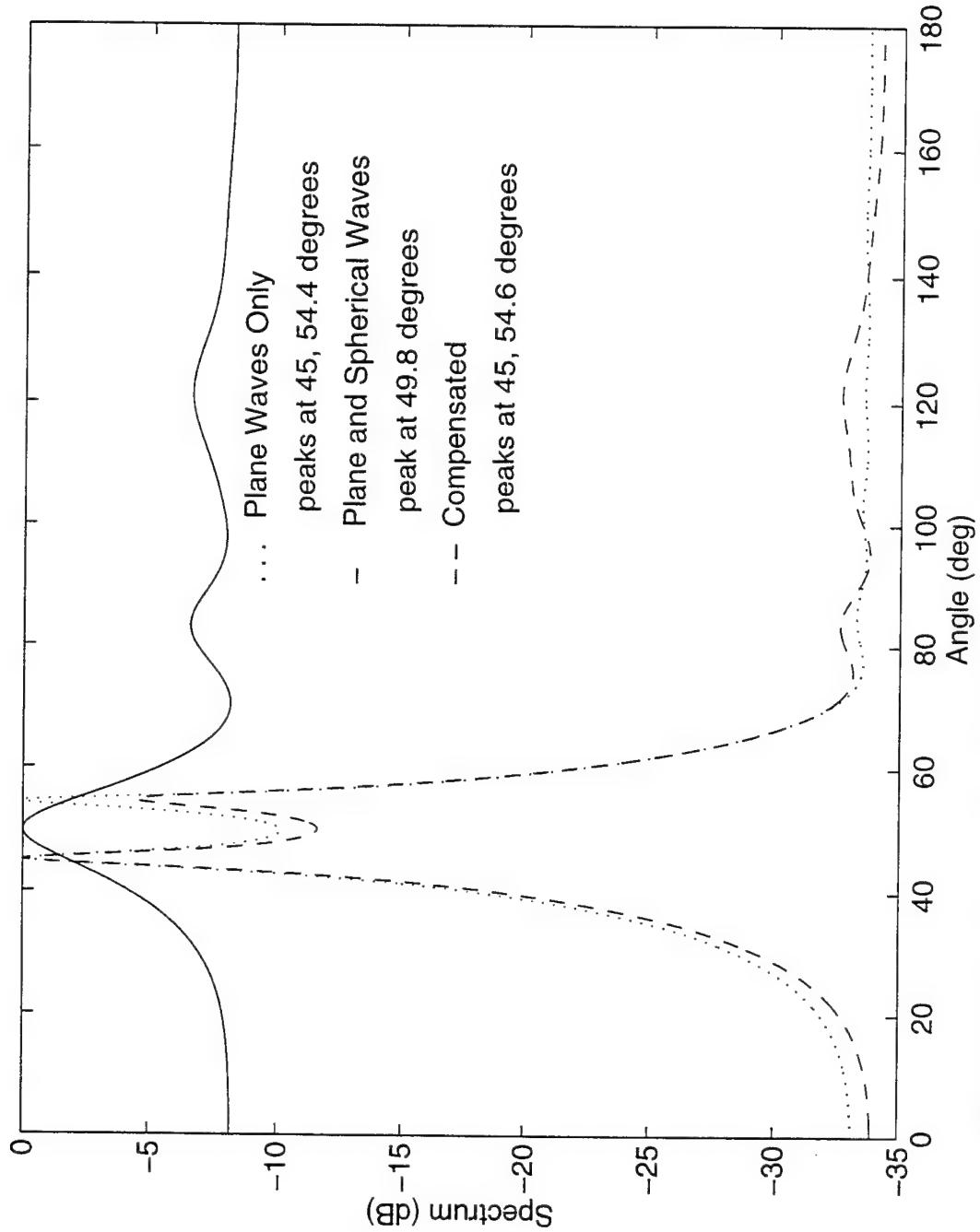
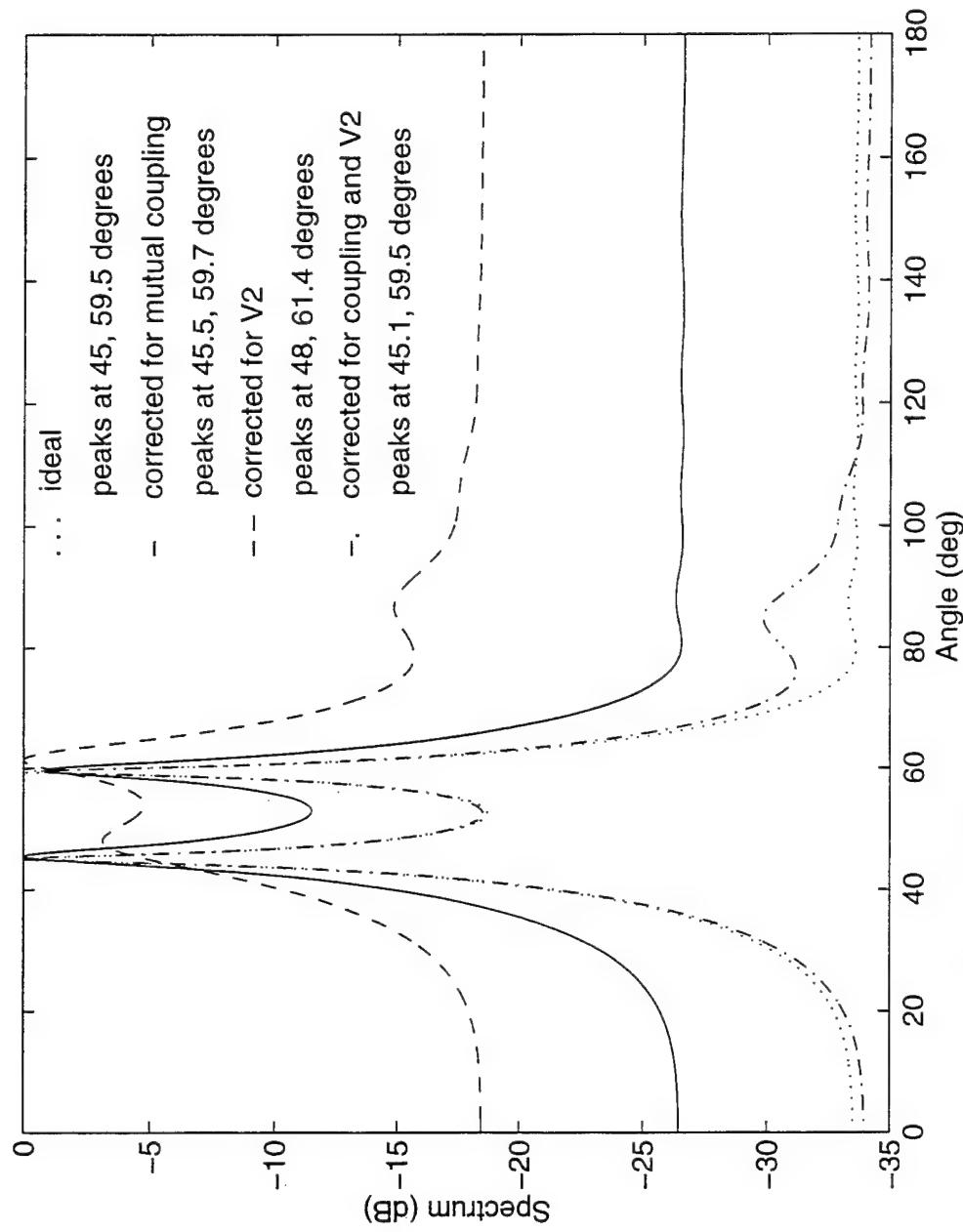
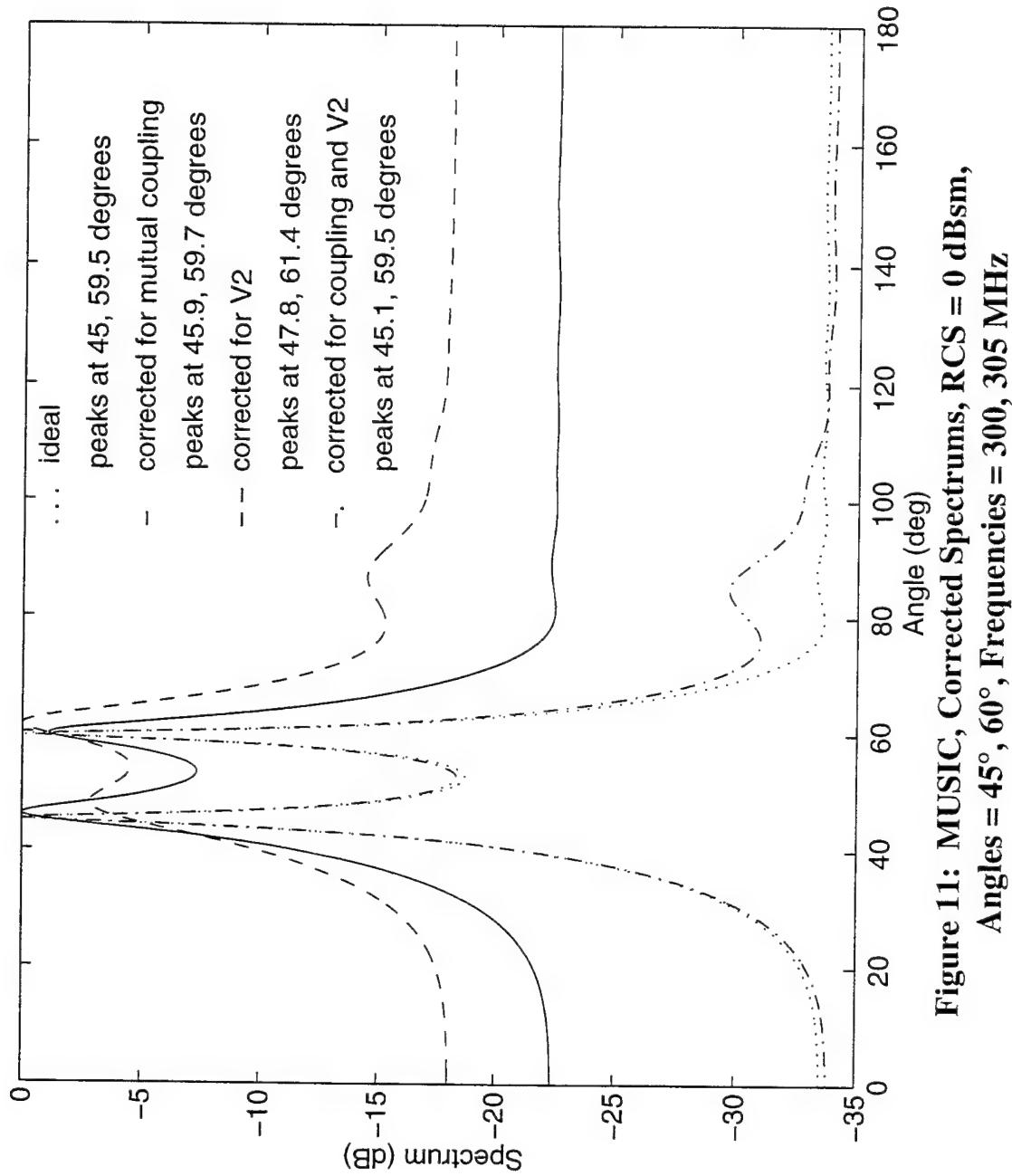


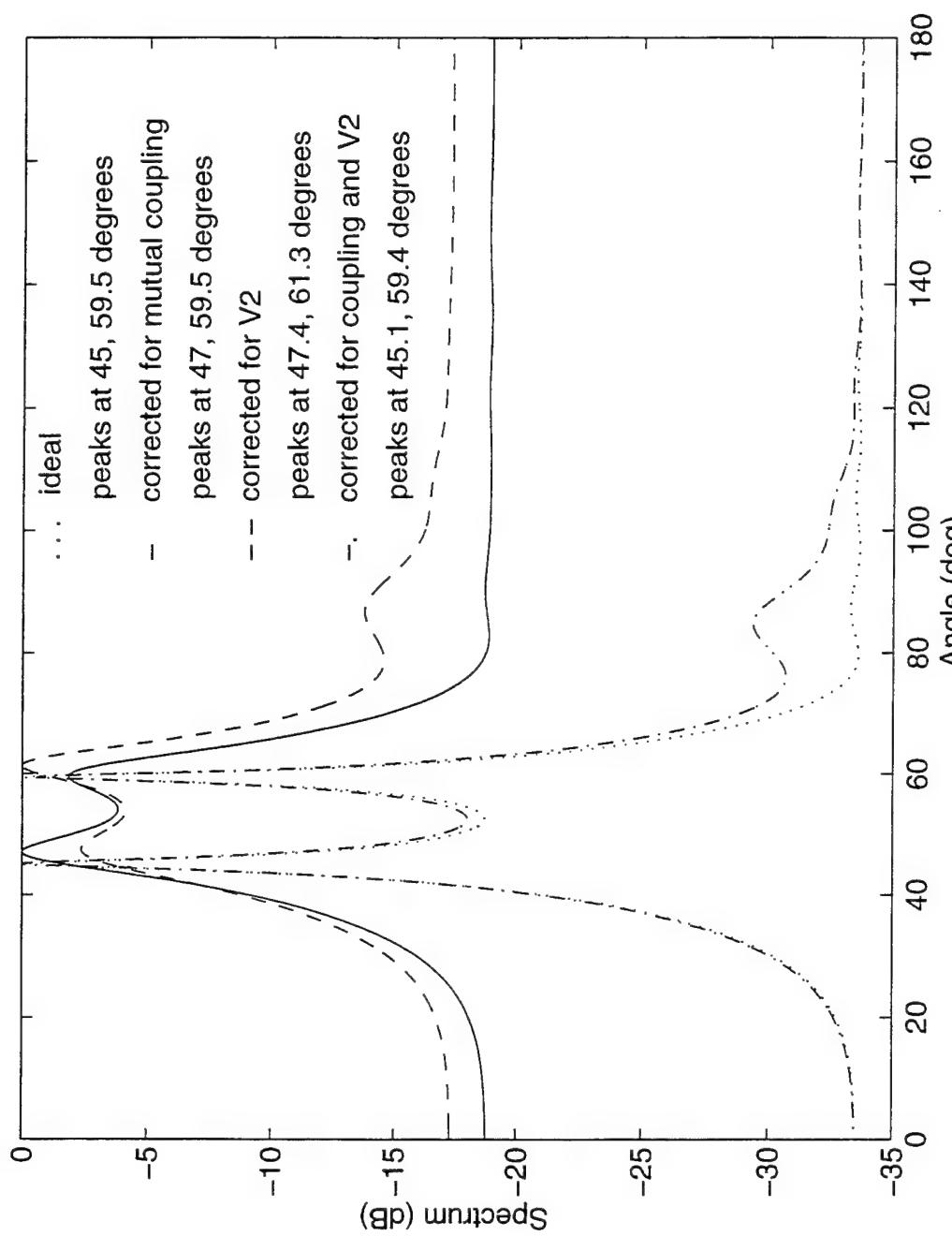
Figure 9: Compensation for a Point Source, $S/I = 0$ dB, SNR = 10 dB



**Figure 10: MUSIC, Corrected Spectrums, RCS = -5 dBsm,
Angles = 45°, 60°, Frequencies = 300, 305 MHz**



**Figure 11: MUSIC, Corrected Spectrums, RCS = 0 dBsm,
Angles = 45°, 60°, Frequencies = 300, 305 MHz**



**Figure 12: MUSIC, Corrected Spectrums, RCS = 5 dBsm,
Angles = 45°, 60°, Frequencies = 300, 305 MHz**

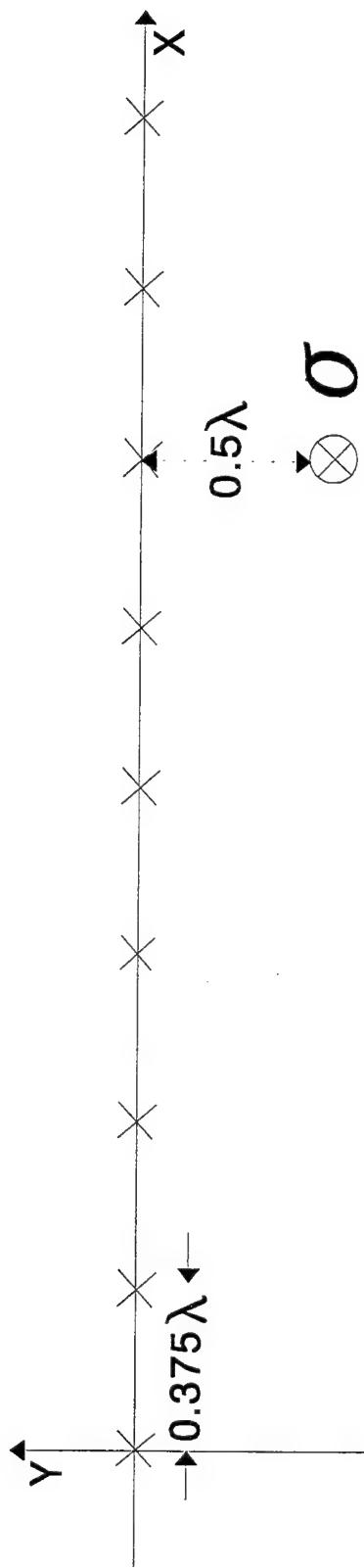


Figure 13: Array with Nearby Scatterer

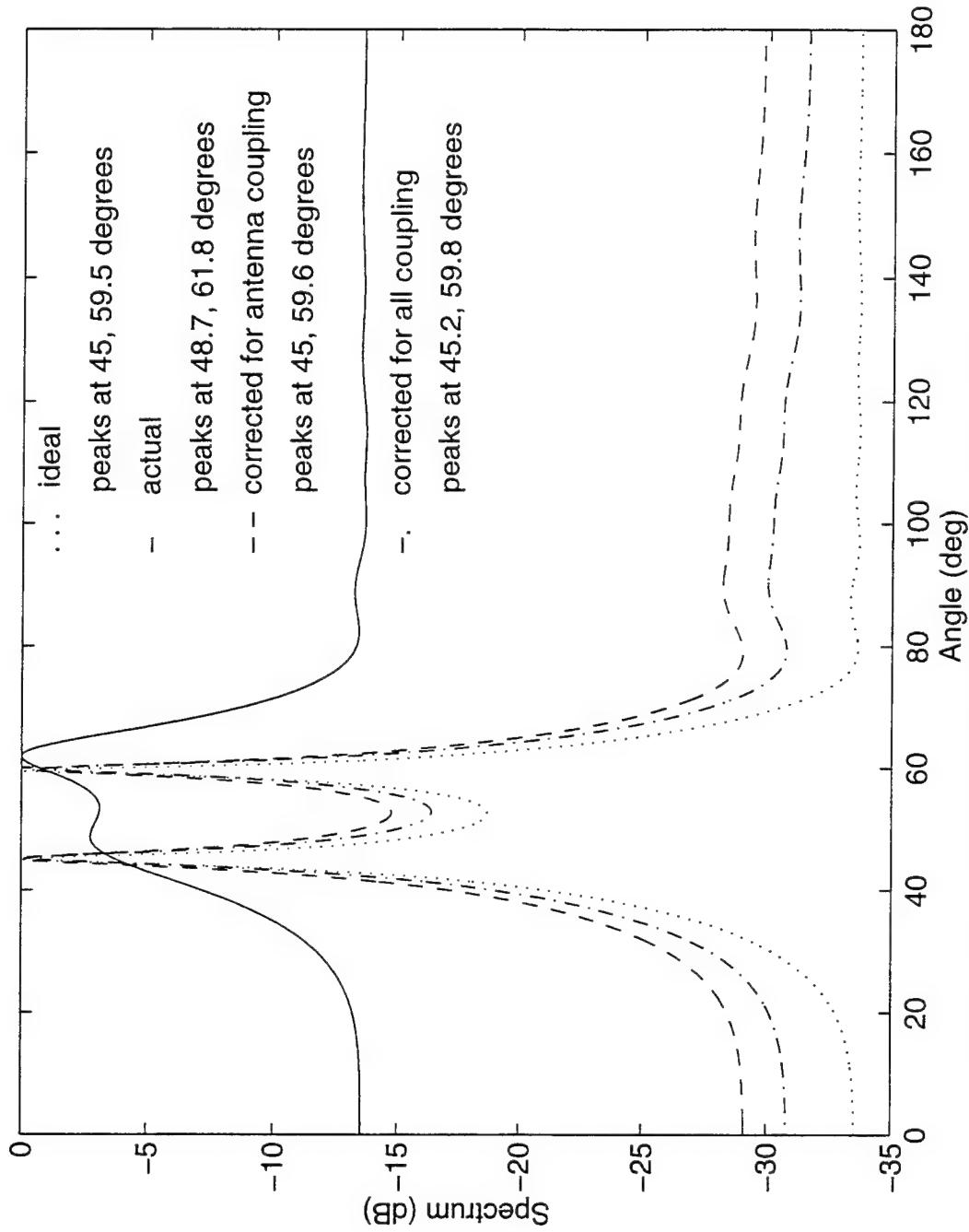


Figure 14: Compensation for a Strong Near Field Scatterer

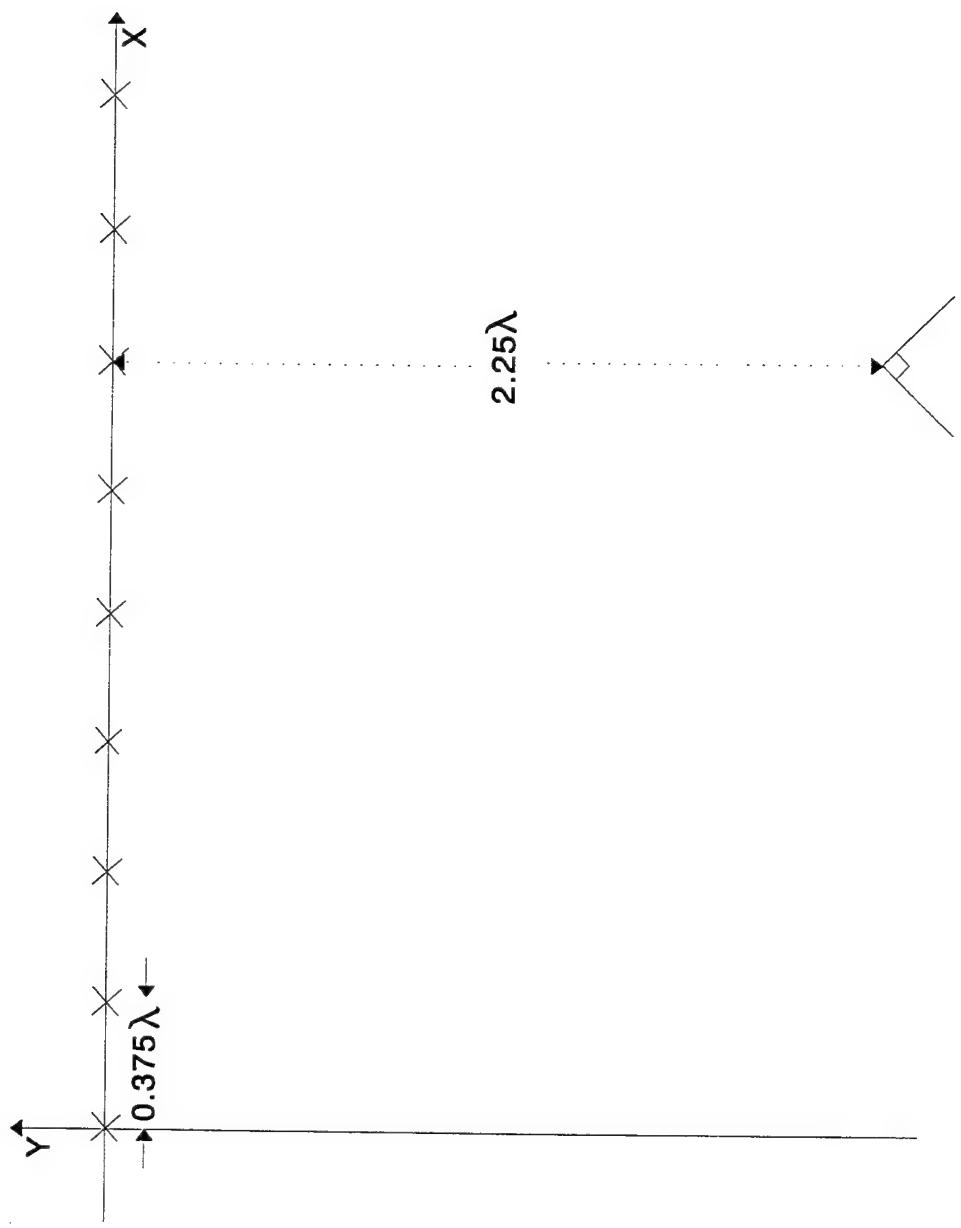


Figure 15: Distributed Near Field Scatterer

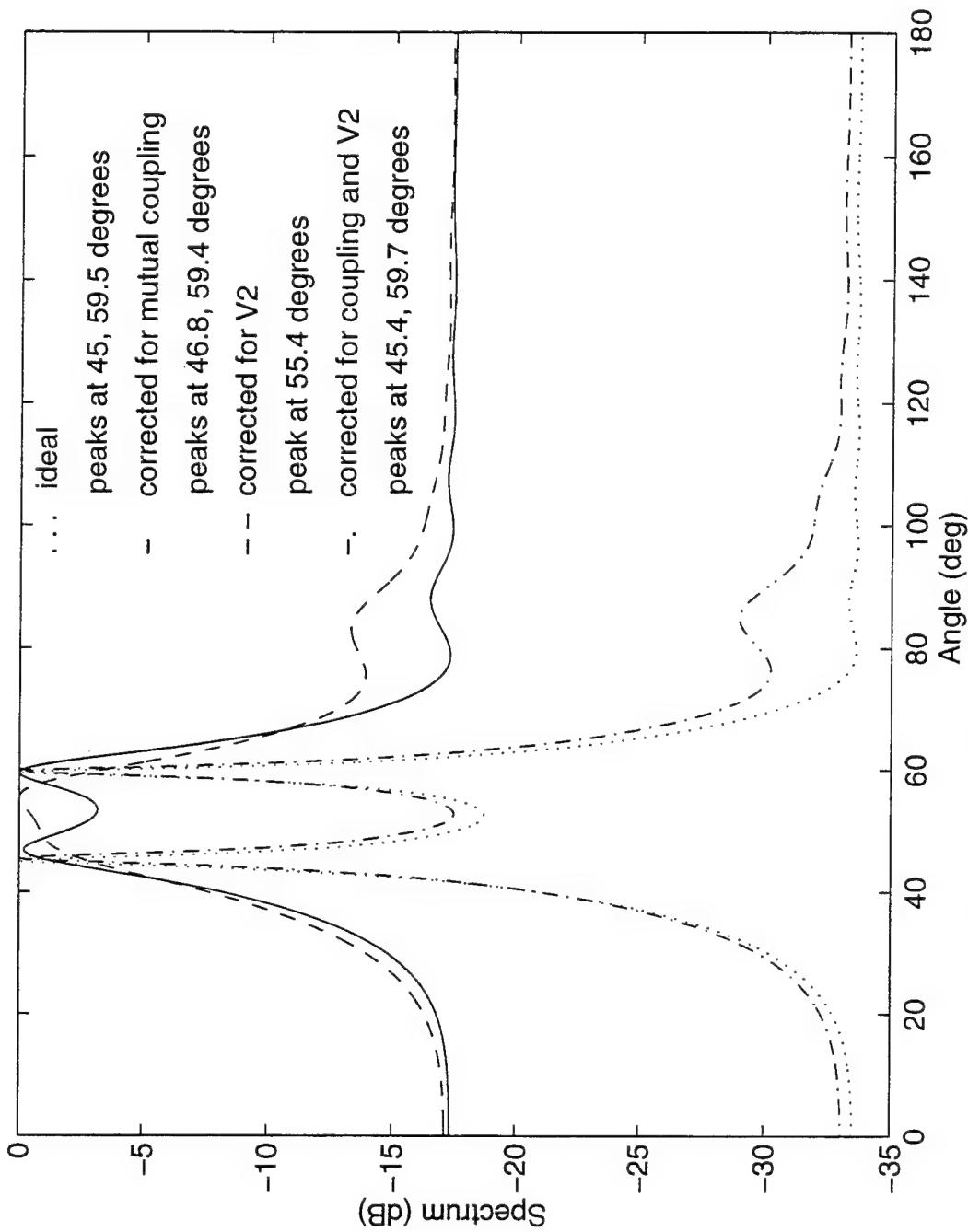


Figure 16: Correction for a Nearby Edge

MULTISTRATEGICAL FUNCTIONAL DECOMPOSER

Marek A. Perkowski

Portland State University,
P.O. Box 751
Portland, Oregon, 97207-0751
mperkows@ee.pdx.edu

Final Report for:
Summer Faculty Research Program
Wright Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Wright Laboratory
December 28, 1995

Marek A. Perkowski
Professor
Department of Electrical Engineering,
Portland State University
mperkows@ee.pdx.edu

Abstract

There are several important combinatorial partial problems that must be efficiently solved in order to develop a functional decomposer for strongly unspecified Boolean functions. They include: Variable Partitioning Problem, or selection of the Bound and Free sets of variables; Column Minimization Problem, or how to combine columns of the decomposition table in order to minimize the multiplicity index; Encoding Problem, or how to encode the combined columns in order to minimize the complexity.

We created an environment, called MULTIS (MULTI-strategy decomposer), that allows to develop various partial decomposition programs, representations, and decomposers, as well to compare them. In this report we provide the information only about the current version of MULTIS. More information on the background theory and future variants can be found in the following additional reports:

1. M. Perkowski et al., "A Survey of Literature on Function Decomposition, Version IV",
2. M. Perkowski et al., "Unified Approach to Functional Decompositions of Switching Functions",
3. M. Perkowski et al., "Development of Search Strategies for MULTIS",
4. M. Perkowski et al., "Documentation of Program MULTIS".

MULTISTRATEGICAL FUNCTIONAL DECOMPOSER
Marek A. Perkowski

1 General Description of the Main Program MULTIS

1.1 General Characteristics

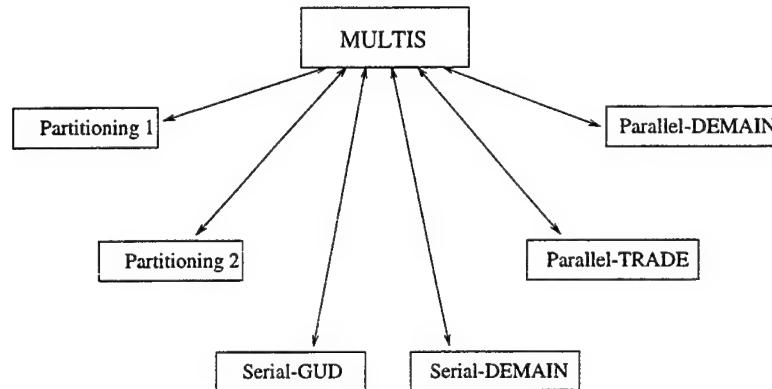


Figure 1: MULTI-Strategy decomposer(MULTIS)

MULTIS is a set of decomposition strategies, each performing one step decomposition on a Boolean function. The input Boolean function as well as the result of the decomposition are in Espresso format so that the result of one decomposition step can be used as an input to the next decomposition step. A list of blocks to be decomposed is kept by MULTIS and each time when a decomposition strategy is called, a block from the top of the list is removed and passed to the next strategy (if it needs to be decomposed). A block doesn't need to be decomposed if its numbers of inputs and outputs are smaller than the current numbers of CLB inputs and outputs. This is automatically detected by MULTIS and such blocks are put into the list of final blocks of the decomposition. The result of one step decomposition is written to disk files, each block to a separate file in Espresso format. File names consists of the string "oooo" followed by an integer(the next consecutive file number) and an ".esp" extension.

At each decomposition step the number of CLB inputs and outputs can be specified or default values used (values from the previous step). The decomposition process is recorded in a log file. The log file name consists of the initial file name with a ".log" extension. The result of the decomposition is written to file result.blif which is the blif format description of the final result. After the decomposition is done the evaluation of the result is performed. The evaluation consists of the following parameters: (1) DFC, (2) number of ones, (3) number of zeros, (4) number of rows, (5) number of CLBs. The result of the evaluation is printed to the screen as well as to result.dfc file . Selecting Main menu Quit option interrupts the decomposition process and blif format result file is created which contains all the so far created blocks.

2 Main Menu

Contains the following options:

1. Variable partitioning 1

Variable partitioning 1 implements Wan Wei algorithm ([?]) and works for single output functions only.

2. Variable partitioning 2

Variable partitioning 2 algorithm is based on the number of don't cares and works for single output functions only. The procedure selects "bond set" number of variables which correspond to the largest number of DCs and which correspond to the smallest number of DCs as the candidate partitions. It is repeated for both ON and OFF-set. As a result, four candidate partitions are selected.

3. Parallel decomposition 1

Parallel decomposition 1 splits a multi-output function into a set of single output functions each having the same set of input variables as the initial function. Number of functions is equal to the number of outputs of the initial function.

4. Parallel decomposition 2 - Luba.

5. Serial decomposition 1 - Luba.

6. Serial decomposition 2 - GUD.

7. Quit.

3 Output Messages

- "Decomposition done :-) ... result in result.blif file"

Decomposition is finished and the result of decomposition is in file result.blif in blif format.

- "Statistics calculation, wait ..."

The message is printed after the decomposition is done and indicates that the result of decomposition is being evaluated. The following parameters are calculated: DFC, number of ones, number of zeros, number of rows, number of CLBs.

- "file to be decomposed next:"

prints the name of the next block (file) from the list of blocks to be decomposed and its number of inputs and outputs.

- "Partitions:"

prints a list of partitions which was created by partition procedure.

- "List of partitions"

prints a list of partitions which was created by partition procedures and prompts the user to select one.

- "block of vacuous variables"

prints the name of file containing a block which all input variables are vacuous. The block is removed from the result.

- "block doesn't need to be decomposed further"

indicates that the block currently being processed has number of inputs and outputs smaller than required maximum number of CLB inputs and outputs and doesn't need to be decomposed further.

- "decomposition successful"

is printed when a block has been successfully decomposed.

- "decomposition doesn't exist"

is printed when a decomposition procedure is not able to decompose the block for a given number of CLB inputs and outputs.

- "Default CLB of inputs and outputs are:"
prints the current values of the number of CLB inputs and outputs required for the decomposition.
- "Enter number of CLB inputs, or press Return to use default:"
prompts the user to enter the number of CLB inputs to be used for the current decomposition.
- "Enter nr of CLB outputs, or press Return to use default:"
prompts the user to enter the number of CLB outputs to be used for the current decomposition.

Besides the above messages, each procedure called from the main menu may print its own messages. They are described in sections related to these procedures. For an example decomposition see the complete documentation.

4 Decomposition strategy DEMAINT

DEMAIN is a functional decomposition program which was designed to be a prototype version of a logic synthesis system. The theoretical basis of DEMAIN is based on a number of papers by Luba[4][5][6][7][8]. DEMAIN has been modified so that it can be incorporated into a larger testing program with additional decomposers and additional options. The larger testing program referred to is the main decomposition program "MULTIS". The additional options available are controlled through the main program "MULTIS". One of the additional options available is that a user may select (from MULTIS) one of two separate partitioning algorithms or may allow DEMAIN to use its own partitioning strategy (provided DEMAIN was selected to decompose a function at the given level). For more specifics on the options available see the section describing the main program of MULTIS. There are two primary components in DEMAIN. These are serial and parallel decomposition.

4.1 Serial Decomposition:

The serial decomposition of DEMAIN can be characterized as a Curtis like serial decomposition with an Ashenhurst like serial decomposition as a special case. Recall that in a Curtis like decomposition the number of outputs of the predecessor block (also referred to as the G block) must be less than the number of inputs to that block. Whereas in the Ashenhurst like decomposition the number of outputs is always one regardless of the number of inputs to the G block. The serial decomposition consists of three sub-components: (1) partitioning of variables into the free and bound set, (2) determining column compatibility, (3) encoding.

The partitioning of variables into the free and bound set is done by starting with a seed partition, having the number of variables equal to the number of inputs for the predecessor block, and exchanging one variable for each new partition tried. This is continued until a decomposition is found or until all partitions of the specified number of inputs have been exhausted. A decomposition is found for a given partition if the encoding is possible for the number of outputs specified by the user.

Finding column compatibility is done by building a compatibility graph from pairs of compatible columns (or blocks). Once the compatibility graph has been built (an edge in the compatibility graph corresponds to a pair of blocks which were found to be compatible), a process of building all maximum cliques is performed. Once complete, the minimum number of maximum cliques are chosen which cover all the blocks in the bound set (i.e. partition P_{i_B}). This minimum number of maximum cliques forms the cover set P_{i_G} which is passed on to be encoded.

4.2 Parallel Decomposition:

The Parallel Decomposition of DEMAIN splits a multi-output function into two new functions. Unlike TRADE, DEMAIN does not split up all outputs so that each new function has only one output. Also different is that the new functions created by DEMAIN do not typically share all the same inputs (i.e. one of the two new functions may have inputs X_1, X_3, X_4, X_5 while the other function may have inputs X_1, X_2, X_5). None of the outputs of one of the new functions is also an output of the other.

For a detailed description of the algorithms used in DEMAIN, refer papers by Luba[4][5][6][7][8].

5 General Universal Decomposer(GUD)

5.1 Basics of GUD

The General Universal Decomposer(GUD) is a functional decomposition program which is capable of decomposing binary and multi-valued, single output and multi output functions. GUD is capable of working as a decomposer by itself but it has been modified so that it can be incorporated into a larger testing program with additional decomposers and additional options. This main program is called MULTIS. Since GUD is a part of Multis and the way of communicating between the different decomposers in Multis is through Espresso files, hence the input boolean function as well as the result of decomposition are in Espresso format. GUD uses two types of data structures BDD's and cubes. The input data is read into BDD's and cubes, and the BDD's are used for the various steps of the decomposition. The cubes are used for encoding purposes and for printing the intermediate Espresso files and the final blif file of the decomposed function. GUD has its own Espresso to blif converter, which is used to print a blif file of the decomposed function. The advantage of this is that the GUD output can be directly verified. This verification is done by using the SIS verifier. GUD has three algorithms for column compatibility, which can be chosen by the user. These algorithms are: (1) Clique Covering, (2) Graph Coloring, (3) Graph Coloring using domination. GUD also has three encoding algorithms, which are: (1) Dichotomy encoding, (2) Dichotomy encoding with cover set selection, (3) Binary encoding. For a general understanding of the flow of control in the program GUD, see the flow diagram in Figure 2.

5.2 INPUT DATA FILE FORMAT

The reader program of GUD is called init_read_input and is located in file gud_serial.c. The input format uses the same input format as in the Espresso program. Some of the input commands in Espresso do not have any effect on the program and new commands have been added. Each of the following subsections describe the different file format commands.

Input Data File Line Types

There are three types of input lines in the input file: (1) A comment line begins with the # symbol at the beginning of the line. (2) A . at the beginning of the line is used to indicate that this is a command line. The different commands are given later. (3) A line that does not start with a # or . is assumed to be tabular data. If a row of tabular data is too big to fit on one line a back slash symbol is used at the end of the line to continue the line to the next line and can be used as many times as required to finish a row of tabular data.

.type Command, Logical Interpretation type, Optional

In .type f the only outputs in the input file are 1 and the rest of the possible terms are interpreted as being 0.

In .type fd the only outputs in the input file are 1 and don't cares, and the rest of the possible terms are interpreted as being 0.

In .type fr the only outputs in the input file are 1 and 0, and the rest of the possible terms are interpreted as being 0.

In .type fdr the outputs in the input file are 1, 0, and don't cares.

The default for gud is fdr.

An _ in the outputs in the input file are interpreted as no output.

.i Command, No. of Inputs, Required

The .i "number" command is used to tell the program the number of input columns in the tabular data. This command needs to be before the .o command in the input data file. The column numbering starts with zero as the first column that is the leftmost column.

.o Command, No. of Outputs, Required

The .o "number" is used to tell the program the number of output columns in the tabular data. This command needs to be after the .i command in the input data file. The "number" will correspond to

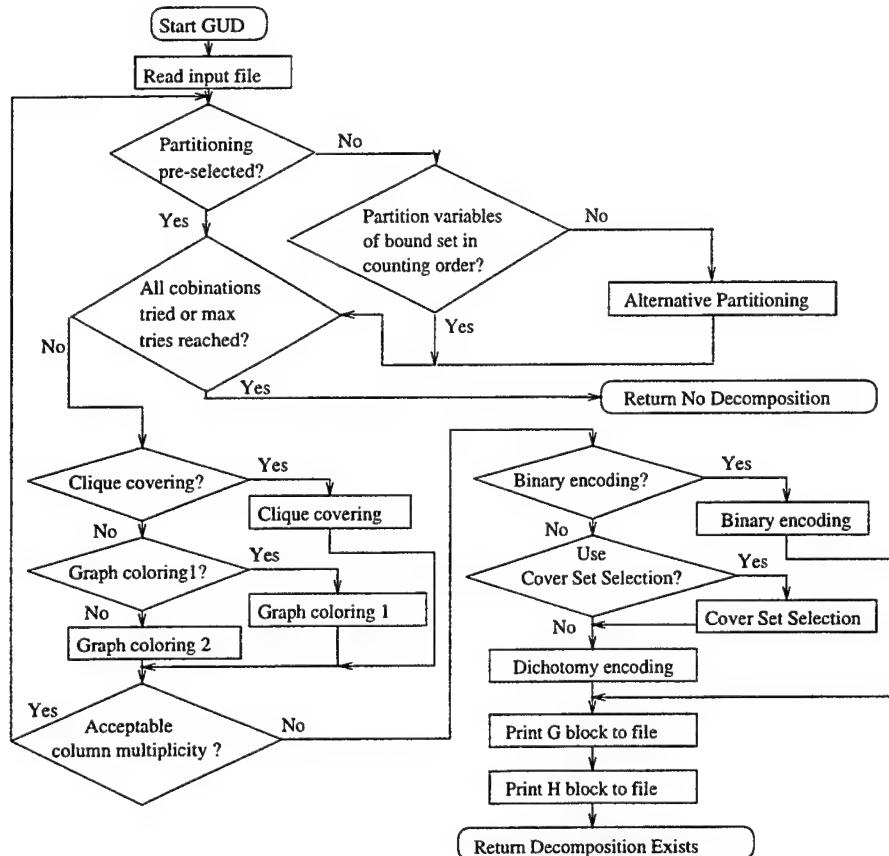


Figure 2: Flow Diagram for GUD

the number of output columns which will start after the input columns. The total number of columns in the tabular data is the number of input columns plus the number of output columns.

.ilb Command, label(s), Required This command needs to be after the .i command. This command reads in the input labels. The number of input labels needs to be the same as the number of input columns.

.ob Command, label(s), Required This command needs to be after the .o command. This command reads in the output labels. The number of output labels needs to be the same as the number of output columns.

.a Command, Free Set columns, Optional

The .a "column number" followed by a space "column number" and as many columns in the free set. If the number of free set columns can't fit on the line additional .a command line or lines can be used.

.b Command, Bound Set columns, Optional The .b "column number" followed by a space "column number" and as many columns in the bound set. If the number of bound set columns can't fit on the line additional .b command line or lines can be used.

.c Command, Set columns, Optional

The .c "column number" followed by a space "column number" and as many columns in the shared variable set. If the number of the set columns can't fit on the line additional .c command line or lines can be used.

.mv Command, Multi-values Size, If Multi-valued Data

The .mv "number" command tells the program the size of the multi-valued data. The "number" is

the number of multi-value values. If the "number" is 3, the multi-value variables can have values 0, 1, or 2. If there is multi-valued data of different sizes the largest size needs to be given.

.mvc Command, Multi-valued Column Numbers, If Multi-valued Data

The .mvc "column number" followed by a space "column number" and as many columns that are multi-valued. If the number of multi-valued columns can't fit on the line an addition .mvc command line or lines can be used. A multi-valued don't care "-" is the same as 0, 1, 2 for 3 multi-valued logic.

.e or .end Command, End of File, Optional The .e or .end command is used after the last line of tabular data to tell the program that this is the end of the tabular data. If this command is not used and the EOF (end of file) is encountered the program will end the reading of the tabular data.

5.3 INPUT TABULAR DATA

The tabular data can either have spaces between the entrees or no spaces. Spaces between the entrees are used for the ease of reading the table. The input data are numbers or don't cares. Don't cares can be represented by -, X, or x. This is shown in the example of an input file.

Example input:

```
# This is an example of an input file.  
.type fdr  
.i 4  
.o 1  
.lb i1 i2 i3 i4  
.ob o1  
.a 0 1  
.b 2 3  
0X10 1  
10X1 0  
010X 1  
0101 -  
1X00 1  
0101 0  
X101 0  
X1X1 0  
.end
```

Types of Input Tabular Data. There are three types of input tabular data that the program will accept and they are: (1) Natural Input Coding, (2) Cube Input Coding, (3) Standard Input Coding.

Natural Input Coding. In natural coding the input data numbers that are the middle four columns a, b, c and d are the same value as the row numbers in binary notation. The input binary data numbers are the middle four columns in Table 1. The middle four columns and output column f is the input data that is read in from a file and the row numbers are assigned to each row by the program. This is shown in the example of natural input data and Table 1. Let us observe that the numbers of rows are not in order, the last row has number 15. Since the numbers of rows are not read by the reader, the reader program creates numbers of rows from the bits of the cubes read by it.

Example of natural input data file:

```
.type fdr  
.i 4  
.o 1  
.lb a b c d  
.ob z  
0000 -  
0001 1
```

No.	a	b	c	d	f
0	0	0	0	0	-
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	-
15	1	1	1	1	-

Table 1: Natural Input Table with Coded Row Numbers.

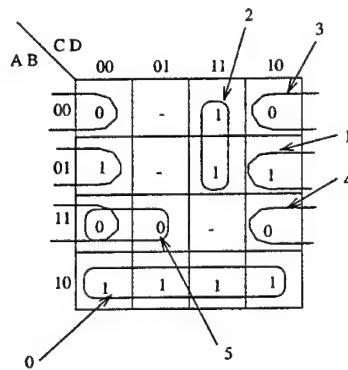


Figure 3: Cube Input Map

```

0010 0
0011 0
0100 1
0101 1
0110 -
1111 -
.end

```

Cube Input Coding. In cube coding the cubes are used for the input with the input row number corresponding to each input cube. Each cube is numbered and is represented by a row in the Table 2. The set of cubes corresponding to the tabular data of Table 2 is shown below in Figure 3. Each row is read in and a row number is assigned as shown in the first column in Table 2. In this case, the reader program assigns a new number to every new row read. It starts counting from zero.

Standard Input Coding. In standard coding the row assigned numbers are not the same as the

No.	a	b	c	d	f
0	1	0	X	X	1
1	0	1	X	0	1
2	0	X	1	1	1
3	0	0	X	0	0
4	1	1	X	0	0
5	1	1	0	X	0

Table 2: Cube Input Table with Coded Row Numbers.

No.	a	b	c	d	f
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	1	0
3	0	1	0	1	0
4	1	0	0	0	1
5	1	0	1	1	1
6	1	1	0	1	0
7	1	1	1	1	1

Table 3: Standard Input Table with Coded Row Numbers.

middle four columns a , b , c , and d , in binary form, as in natural coding and are not input data cubes in cube coding. The row numbers in the first column of the Table 3 are assigned while each row is read in, the middle four columns, and the output column f is the input data values that is read in from the input data file. Like in the last subsection, in this case, the reader program assigns a new number to every new row read, starting from zero.

Standard coding is used for multi-valued data with each row corresponding to an assigned row number. The input values can be mixed multi-valued and binary. A multi-valued example using Standard Coding is next.

```

Multi-valued Example with Standard Input Coding .type fdr .i 4
.o 3
.lb mv1 mv2 bi3 bi4
.ob outmv1 outmv2 outmv3
.mv 3
.mvc 0 1 4 5 6
 0,1   0      X   X   0     1      X
 1,2   0,1    0     0   0     1      0,1,2
 0,2   0,2    X     0   0,1  0,1,2  0,2
 2     1,2    1     1   0,2  1       2
 0,1   0,1,2  X     1   2     2     1
.end

```

This function has four inputs and three outputs. They are all three-valued. X means any value. $0,2$ in output means that either value 0 or value 2 can be taken as a value of the output variable. This is an example of a function with the Generalized Don't Cares.

5.4 GENERALIZED DON'T CARES FOR MACHINE LEARNING

A machine learning program can be realized by a decomposition function that will accept input that is multi-valued and has multi-valued outputs. The program is given inputs of different categories and the outputs are the results of the program.

An example of output categories is: chair = 0, table = 1, desk = 2. The entries that belong to more than one output value(or class) are called *Generalized Don't Cares*. The following are examples of *Generalized Don't Cares*: chair or table = 0,1; chair or desk = 0,2; table or desk = 1,2; chair, table or desk = 0,1,2 = X . An example of a multi-valued-input, multi-valued-output, three-output function is shown in Table 4. Each row number is assigned while reading in a line of data.

Output partitions.

$$\Pi_{F_1} = \{B_0; B_1; B_2\} = \{\overline{0, 1, 2, 3}; \overline{2, 4}; \overline{3}\}$$

$$\Pi_{F_2} = \{B_3; B_4; B_5\} = \{\overline{2, 4}; \overline{0, 1, 2, 3}; \overline{2}\}$$

$$\Pi_{F_3} = \{B_6; B_7; B_8\} = \{\overline{0, 1, 2, 4}; \overline{0, 1}; \overline{0, 1, 2, 3}\}$$

When two variables are consistent they can be combined to a single new variable. Two variables

No.	W	X	Y	Z	F_1	F_2	F_3
0	0,1	0	X	X	0	1	0,1,2
1	1,2	0,1	0	0	0	1	0,1,2
2	0,2	0,2	X	0	0,1	0,1,2	0,2
3	2	1,2	1	1	0,2	1	2
4	0,1	0,1,2	X	1	1	0	0

Table 4: Generalized Multi-Valued Input Table with Coded Row Numbers.

W & X	Y & Z	F_1	F_2	F_3
0	X	0	1	0,1,2
1	0	0	1	0,1,2
0,2	0	0,1	0,1,2	0,2
2	1	0,2	1	2
0,1	1	1	0	0

Table 5: Combined W and X Multi-Valued columns, Y and Z binary columns

are said to be consistent they if they have no contradictions in every row(cube) of the table. The multi-valued columns W and X , and the binary columns Y and Z can be combined. This is shown in the Table 5.

6 VARIABLE PARTITIONING OF BOUND, FREE, AND SHARED SETS OF INPUT VARIABLES

6.1 Variable Partitioning Methods In GUD

There are two variable partitioning methods that GUD uses if a free and bound set is not given from MULTIS(or from the input file). The first method is referred to as the "CLB size partitioning" The second method is referred to as an "alternative partitioning method" for lack of a more suitable name at this time. Both of these partitioning schemes are quite simple and fast as they select partitions in a psuedo random fashion.

CLB size Partitioning Method. The number of variables in each partition selected is the same as the number of inputs chosen by the user. This number is equal to the number variables in the bound set(also referred to as the CLB size). Partitions of variables are selected in counting order to try and find a decomposition. If after all the combinations of that CLB size has been tried, or after the limit of MAX_TRIES=100 has been reached, GUD will return no decomposition. Of course the limit can be changed if so desired. Changing MAX_TRIES would entail changing the value in the source file and recompiling it. If no decomposition was found using this partitioning approach then the user may try decomposing the same function block using a different partitioning method or may select a different decomposer or one of the other options from the main menu of MULTIS.

Alternative Partitioning Method. This method starts with the first two input variables as the bound set. If this bound set does not create a valid decomposition then the first three input variables is taken. This process of adding one more input variable to the bound set continues until the bound set size is equal to half the total number of input variables.

If no decomposition has been found from the previous partitions selected, then the last two input variables are chosen as the bound set. If this bound set does not create a valid decomposition the last three input variables are taken next. This process of adding one more input variable continues on until the bound set size is equal to half the total number of inputs.

Next the middle two input variables are chosen for the bound set. If this bound set does not create a valid decomposition then another input variable next to the two middle input variables is added to

the bound set. This process of taking one more input variable continues until the size of the bound set becomes half of the input variables.

If a decomposition has still not been found the bound set is created by using a random number generator. The random number bound set size is any number of inputs up to half the total number of input variables. Each time a partition is tried a counter is incremented. If the count exceeds MAXTRIES=100 GUD returns no decomposition to the main program MULTIS. If no decomposition was found using this partitioning approach then the user may try decomposing the same function block using a different partitioning method in GUD or may select a different decomposer or one of the other options from the main menu of MULTIS.

7 GRAPH COLORING APPROACH TO COLUMN MINIMIZATION

The graph coloring approach to column minimization uses an incompatibility graph instead of a compatibility graph as is used in set covering and in maximum clique approaches. There are significant advantages to using a graph coloring approach vs. the other mentioned approaches. Graph coloring requires less memory than set covering, and good polynomial time heuristic algorithms exist for its solution for large input sets. GUD has two graph coloring algorithms which will produce the necessary partitioning of the bound set into compatible classes (Π_g). The first is a fast heuristic algorithm written by Wan Wei, the same one as used in TRADE[13]. The second is a newly created exact algorithm, designed by Perkowski[see a new report by Roger Shipman]. Each is contained in separate file with an associated header file.

7.1 Graph Coloring approach based on "color influence"

Wan Wei's method, which he calls color influence graph coloring, is contained in the files `color_influence.c`. The entry point, which is called from `gud_serial.c`(the main dispatching source file for GUD) is

`color_influence_decompose()`. This interface routine first creates an incompatibility graph, then calls the routine `graph_coloring()`, a reimplementation of Wan Wei's pseudocode as given in his master's thesis Wan[13]. The pseudo-code is repeated in the source file. All routines except the interface routine are declared static, only callable from within this file.

When `graph_coloring` returns, an alternate data structure, originally created for use by the set covering decomposition, is created from it, and the incompatibility graph is disposed. The alternate data structure "G", called a clique cover, is returned. Also, the function return value is the number of colors used to color each partition block of G. If this number is too large (greater than 31), then the value `TOOMANYCOLORS` is returned. In this case, a clique cover is not created.

7.2 Graph Coloring approach based on "dominating nodes"

The second approach, created by Perkowski[], is a previously unimplemented approach. The main function is called `graph_covering_decompose()`. The file containing this function is "`graph_covering.c`". The algorithm works by observing that, within the incompatibility graph, any node whose edges are a subset of another node's, can be colored with its same color. This can be expressed symbolically as:

If `edge(A) <= edge(B)`, then `color(A) <- color(B)`.

A node so dominated is removed with all its edges. This will perhaps reveal other nodes which are also dominated. The process of locating a dominating node and removing it continues until a cycle is discovered, or until a complete graph is obtained. Once a complete graph is obtained, each node will receive a different color, and each of the dominated nodes can be colored according to the assignments made. If a cycle is found, the node with the largest number of edges is chosen, and all the maximal independent sets which include that node are selected as branching choices, to be tried one after the other. Choosing the node with most edges is a heuristic; it implies the least number of compatibilities. This is hoped to effectively cut the cycle so that the domination process can continue. According to

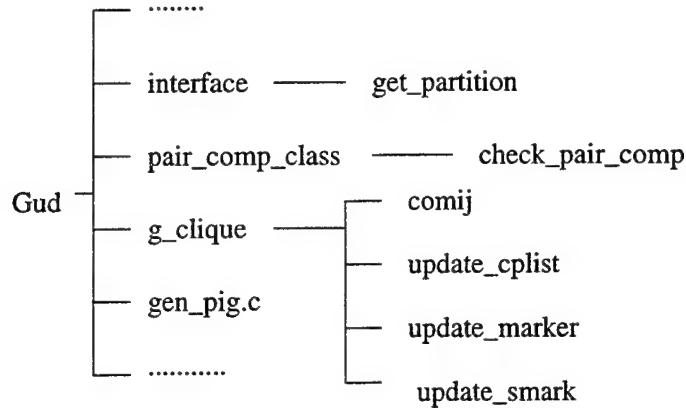


Figure 4: Subroutine calling diagram of Clique approach to Column Minimization

Swamy[12], a subset S of the set of nodes of a graph, N , is a maximal independent set (MIS) if and only if the set $S^* = N - S$ is a vertex cover: therefore no node not in the MIS can be the same color because each one will have an edge with some other node in the MIS.

Thus, for each MIS chosen, the next available color is assigned to it (and to each of the nodes it dominates). Those nodes are then removed, and all their edges, and the process continues from the top. Once a coloring is obtained, it is stored, if less than the minimal coloring so far, and the algorithm backtracks to the last cycle, and chooses the next possible MIS. When all possible MIS's have been chosen for all cycles found, the branching choice which produced the optimal coloring is returned.

8 THE MAXIMUM CLIQUE APPROACH TO COLUMN MINIMIZATION

The program for column minimization presented in this section is based on heuristic finding large cliques. It has two parts. The first part is to find the partitions for the, given as argument, bound set, free set, and output set of variables. This is done by subroutine "interface.c". The other part is to find partitions for PI_g, which is done by subroutines "pair_comp_class.c", "g_clique.c", and "gen_pig.c". Fig. 4 is the subroutine calling diagram.

8.1 SUBROUTINE INTERFACE.c

Included in file interface.c:

Function of subroutine interface.c:

Given BDDs of partitions for each input and output variable, generate BDDs of partitions for each: free set A , bound set B , shared set C , as well as BDDs for output partition.

Format of subroutine interface.c:

```

interface(bddm, partition, in_partition, out_partition, Var_A, Var_B, Var_C, a, b, f, an, bn, fn)
bdd_manager bddm;
PARTITION *partition;
int in_partition;
int out_partition;
Node Var_A;
Node Var_B;
Node Var_C;

```

```
bdd **a;
bdd **b;
bdd **f;
int *an;
int *bn;
int *fn;
```

PARTITION is a linked list of pointers, each pointer points to a linked list of BDDs corresponding to the partitions of each variable. Every partition is a set of blocks, every block is represented by a BDD. **partition* is the pointer of *PARTITION* for each input and output variable.

Node is a linked list of the indices of variables in a subset. *Var_A*, *Var_B* and *Var_C* are Nodes corresponding to the variables in free set, bound set and common set.

***a*, ***b* and ***f* are the outputs of interface, they are pointers to array of pointers to BDDs (blocks) for the partitions of free set, bound set and the output set. **an*, **bn* and **fn* are the numbers of elements of the arrays for free set, bound set and the output set.

Algorithm of subroutine *interface.c*:

1. Call subroutine "get_partition" to get partitions (in BDD representation) of the free set of variables. It passes the index-linked list of free set along with the partition linked list to subroutine "get_partition", and gets the array of the pointers to the BDDs (partitions) of this subset.
2. Call subroutine "get_partition" to get partitions of the bound set of variables.
3. Call subroutine "get_partition" to get partitions of the output set of variables.

Function of subroutine *get_partition*:

Given a subset of the input variables, generate the BDDs for the partitions of this subset. Put the pointers to the BDDs into an array. The return values are the pointers to the array and the number of its element.

Format of subroutine *get_partition*:

```
int get_partition(bddm, partition, Y, **y);
bdd_manager bddm;
PARTITION *partition;
Node Y;
bdd **y;
```

**partition* is the same linked list passed to subroutine *interface*.

Y is the linked list for the variables in the subset that needs to be processed.

y is the pointer to the returned array.

Return value is the number of elements in the array.

Algorithm of subroutine *get_partition*

1. Get partition blocks of one variable from free set (or bound set, or output set), put them into *final_lst*.
2. Get partition blocks of another variable from the same subset (free set, bound set or output set), store them in *D_block*.

3. Execute all intersections on BDDs (blocks) from *D_block* with BDDs from *final.lst*, each block with each block. Check if the intersection is zero or included in each other, if yes discard the smaller one, if no - continue. The intermediate result is stored in *temp_lst*, when created from partitions with *n* and *m* blocks, respectively, it can get the maximum of *n * m* BDDs (partition blocks). Pass the final result to *final.lst*.
4. Go back to step 2 until there are no more unused variables in the subset of variables.
5. Go through the *final.lst*, get the pointers to each of its elements, store the pointers in array *y*, which is the array of pointers to the BDDs. Count the number *num* of BDDs in the *final.lst*.
6. Return *num*, the array *y* can be accessed by address of *y*.

8.2 SUBROUTINE PAIR_COMP_CLASS.c

All these is in file **Pcc.c**: This subroutine creates the pairwise compatible classes of columns, which is the same as the Column Compatibility Graph. It does this by checking the compatibility of every two blocks of bound set partition according to Luba's compatibility method. This is done after generating the partitions for the bound set, free set and the output set. The result is then given to *clique.c* to find the Maximum Compatible Classes.

Assumption: Given are arrays of pointers to the partition blocks of free set *A*, bound set *B*, and output set *F*.

Format of the subroutine pair_comp_class:

```
pair_cl *
pair_comp_class(bdd_manager bddm,
bdd *a,
bdd *b,
bdd *f,
unsigned int an,
unsigned int bn,
unsigned int f )
```

where **a*, **b* and **f* are arrays of BDD pointers to bound set *b*, free set *a* and output set *f*; *an*, *bn*, *fn* are the numbers of respective partition blocks (BDD pointers).

Return value is a link list with each element being a pair of compatible blocks.

Algorithm of the subroutine pair_comp_class:

1. Call subroutine "*check_pair_comp*" to check if two blocks in the bound set are compatible or not.
2. If yes then add them to the linked list *cplist*,
3. If they are not compatible then check iterate - another pair. After each pair has been checked, return *cplist* to the main program.

Format of the subroutine check_pair_comp:

```
int
check_pair_comp(bdd_manager bddm,
int i,
int j,
bdd *a,
bdd *b,
bdd *f,
unsigned int an,
unsigned int bn,
```

```
unsigned int fn)
```

where $*a$, $*b$, $*f$, an , bn , fn have the same meaning as in *pair_comp_class*, i and j are the indices of blocks in bound set b to be checked. It returns an integer indicating whether blocks i , j are compatible or not, according to Luba's compatibility checking method.

8.3 SUBROUTINE CLIQUE.c

Function of subroutine clique:

Given a linked list of compatible pairs for a given bound set, find a good maximum compatible class for generating *PIg*. The problem is equivalent to a graph clique problem, each block in bound set can be viewed as a vertex, if two blocks are compatible, there is an edge connecting the two vertices. Finding an MCC is equivalent to finding a maximum clique in the graph clique problem.

Format of subroutine clique:

```
cliq_hl *g_clique(cplist, bn)
pair.cl *cplist;
int bn;
```

where **cplist* is the linked list of compatible pairs generated by subroutine *pair_comp_class*, *bn* is the number of blocks in the bound set, it is the value returned by subroutine *interface*.

Algorithm of subroutine clique:

(1). Create array *globm*[]; each of its elements corresponds to a vertex in the graph, and is used to store the information whether the vertex has been included in a clique or not. The initial value of elements in *globm*[] is set to zero.

(2) Check if all elements of *globm*[] have been marked. If yes, all vertices have been included into at least one clique. If the work is done, then return with *clhl* which is the linked list of pointers to cliques. If not, continue.

(3) For each vertex, count the number of edges connected to it. Find the one which has the highest value. Mark this vertex by setting the respective element in *globm*[].

(4) Store this vertex in *marker*[0].

(5) Find another node which has not been marked.

(6) Check if the new node is compatible with all nodes from the clique by calling subroutine *cmpij*. If yes, the new node is in the clique. Add it to *marker*[]. Call subroutine *update_marker* to mark the corresponding element in *globm*[]. Call subroutine *update_cplist* to update the Compatible Graph by deleting the edges which have been included in this clique. If not, go to step 4.

(7) All nodes in the graph checked? If yes, the vertices in *marker*[] are in one clique. Store this clique in linked list *cl*. Add the pointer to this clique and to the *clhl*, which is another linked list used to store the pointers to the cliques. Update the Compatibility Graph by deleting the edges which have been included in this clique. Call subroutine *update_smark* to change the order of the vertices in *smark*[]. Go to step 2. If not, go to step 5.

8.4 SUBROUTINES CALLED IN G_CLIQUe.c

The following are the other subroutines in file *g_clique.c*.

1. *int cmpij(int, int, pair_cl*)*:

Check if block i and block j are compatible or not. It takes the indices of two blocks and the pair compatible link list as input, check through the list to see if the two blocks is a compatible pair or not.

2. *pair_cl *update_cplist(int*, int, int, pair_cl*)*:

After a vertex is found to be in a clique, the edges which connect it to other vertices in the clique are deleted by calling this subroutine,

Taking the index of the new vertex, the *marker*[] and the pair compatible link list *pair_cl* as input, this subroutine deletes the edges between the vertex and other vertices in *marker*[] by removing the corresponding elements of compatible pairs in *pair_cl*.

3. *update_smark.c*:

smark is used to during handling one clique, to keep track if a node is included into this clique or not.

If it is in this clique, then add it to *smark*. When one vertex has been included in one clique, it is moved to the last element of *smark*[]. With this subroutine, we try to avoid such case that some vertices have more chance to be chosen first just because their index number. This subroutine is called after each time a clique is found. This subroutine is a heuristic.

4. *update_marker.c*:

This subroutine is used to update *globm*[]. This subroutine is used to keep track of all the blocks being included in the cliques. It is called in *g_clique* each time a vertex is included in a clique. Whenever a vertex is found to be in a clique, the corresponding element in *globm*[] is set to 1. When all the elements in *globm*[] are 1's, which means each vertex in this graph has been included in at least one clique, the procedure of *g_clique* stops.

8.5 SUBROUTINE GEN_PIG.c

file gen_pig.c: This subroutine generates partition *PIg* from set covers. After encoding, function *H* is obtained from it.

Format of subroutine gen_pig:

```
bdd *
gen_pig(Clique clhlhead,
bdd *b)
```

where *clhlhead* is a linked list of the heads of the set covers (cliques) from *clique.c* and **b* is an array of BDD pointers to the bound set.

Algorithm of subroutine gen_pig:

- (1) Find out how many cliques exists in *clhlhead*.
- (2) Allocate enough memory to store partition *PIg*.
- (3) For each clique in *clhlhead*
 - execute *bdd_or* operation on its elements. This creates the partition of *PIg*.
- (4) Return *PIg* to the main program to be handled by the printer routine and next iterations of decomposition.

9 COVER SET SELECTION

9.1 Description of Cover Set Selection

"Cover Set Selection" is the process of selecting and assigning columns to optional classes in the cover set *PiG*. For example, if column 3 is compatible with classes A,B and E, then a choice can be made as to which class the column should be assigned to. Assigning a column to more than one class is sometimes possible. However, it is often not possible and it is much more complicated to encode properly. That is why a single choice must be made in order to ensure that a cover set can be encoded and will still maintain consistency between the decomposed function and the original function. As one might guess, making different selections will lead to either more or less complex subfunctions depending on the choices made. Therefore the goal of the Cover Set Selection algorithm is to end up with less

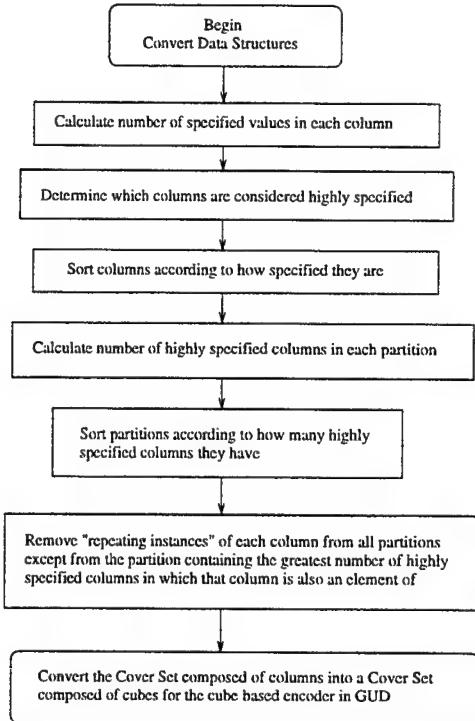


Figure 5: Flow Diagram for the Cover Set Selection Algorithm

complex subfunctions than would result from simple random selections. The basic steps in the Cover Set Selection algorithm are the following:

1. Sort columns according to how highly specified they are
2. Tag a small percentage of the most specified columns (perhaps 1/3)
3. Sort partitions according to how many highly specified columns each has
4. Starting in descending order with the partition which has the greatest number of the tagged columns (highly specified columns), remove all repeating instances of columns from all other partitions.

These steps are reiterated in the flow diagram of Figure 5.

9.2 Comparison of successor blocks for different cover sets selected

At present the subroutine *Cover_Set_Selection* only works with the dichotomy encoder in program GUD because of differences in data structures used by the other encoders. It should be noted that the subroutine *Cover_Set_Selection* will only make a difference in a decomposition when the function (or subfunction) presently being decomposed is highly unspecified. When a function is highly unspecified there may be numerous "optional" assignments of columns to classes. For fully specified functions, *Cover_Set_Selection* will have no effect on the decomposition because there are no optional assignments possible. An example comparison to illustrate the consequence of making different selections can be seen in Figure 6. Shown in this figure is the original function (F) and the successor block (H) for two separate cover sets selected. Below the "H" K-map is a label corresponding to each class in the cover

		cde	F					
ab		0 1 2 3 4 5 6 7						
0 1	-	0 - - - 0 0						
- 0	0 -	- 1 - - -						
0 -	- -	0 0 1 0						
- 1	- 1 0 -	- - 1						
S2 S3 S4	S1 S2 S3 S4	S2 S3 S4	S1 S2 S3 S4	S2 S3 S4	S1 S2 S3 S4	S2 S3 S4	S1 S2 S3 S4	S2 S3 S4

The function F above is shown with the classes each column is compatible with. The classes which are circled are the resulting assignments of columns to compatible classes using the Cover Set Selection algorithm. The resulting K-map for the successor block(H) is shown to the right.

		cde	F					
ab		0 1 2 3 4 5 6 7						
0 1	-	0 - - - 0 0						
- 0	0 -	- 1 - - -						
0 -	- -	0 0 1 0						
- 1	- 1 0 -	- - 1						
S4 S3 S4	S1 S2 S3 S4	S2 S3 S4	S1 S2 S3 S4	S2 S3 S4	S1 S2 S3 S4	S2 S3 S4	S1 S2 S3 S4	S2 S3 S4

		xy	H			
ab		0 1 0 -				
0 0	1 0	0 1 0 -				
0 -	- -	0 0 1 0				
- 1	- 1 0 -	- - 1				
S1 S2 S3 S4						

Shown above is the same function F except this time an arbitrary assignment of columns to compatible classes results in a different K-map for the successor block. It can be observed that more don't cares were preserved in the previous assignment using the Cover Set Selection algorithm.

Figure 6: Comparison of successor blocks for different cover sets selected

set. The number of distinct classes is equal to the minimum column multiplicity(4 in this particular example). Below each of the columns in the F K-map are all classes that each column is compatible with. The classes which are circled are the ones which have been selected to form the final cover. The final cover results from taking the union of all columns assigned to the same class. This final cover forms the columns of the successor block. Note that, by properly selecting which class to assign each column to, more don't cares may be preserved thereby reducing the complexity of the successor block.

9.3 Subroutines used in Cover Set Selection algorithm

All subroutines used in the Cover Set Selection algorithm are located in file "cover_select.c", "print.c" and "print.h". The following shows the list of inputs that are arguments to the main subroutine.

```
void Cover_Set_Selection(New_Pig_Block **cover_set_addr,
                        COLUMN_SET **column_set_addr,
                        int column_mult,
                        int num_rows,
                        int num_columns)
```

- *Cover_Set_Selection()* is the main Subroutine for the Cover Set Selection algorithm. It heuristically selects which partition each repeated column should be assigned to in order to reduce the complexity of the successor block(H). It's primary inputs are two arrays of lists. One of these array of lists(the input cover set) stores a set of partition blocks and an associated list of columns(or blocks) for each partition block. The other array of lists(the set of columns) stores a set of columns(or blocks) and an associated list of minterms(or cubes) for each column. These arrays of lists were converted from the BDD data structures which were used in prior steps in program GUD. The conversion from BDD's to arrays of lists is done by the subroutines in section 9.4. All other subroutines for the Set Selection algorithm are called from within *Cover_Set_Selection()*.
- *Data_Struct_1_Convert()* converts a data structure from an array of lists to an array of arrays to save time accessing specific elements which would otherwise need to be searched for(i.e. lists would need to be traversed to find if an element is a member of a class). Input for *Data_Struct_1_Convert()* is of type (*New_Pig_Block **).

- *Data_Struct_2_Convert()* is the same as subroutine *Data_Struct_1_Convert()* except that the input is of different type. Input for *Data_Struct_2_Convert()* is of type (COLUMN_SET *).
- *sort_descending()* is a subroutine which forms a list of elements from an array and sorts them in descending order according to the size of each element .
- *array_highly_spec_columns()* is a subroutine which returns an array with as many elements as there are columns and which has a value 1 assigned to columns which are determined to be highly specified relative to the other columns. Other elements in the array are assigned zero. The input is a sorted list of column sizes, a decimal number "X"(between 0 and 1), and the number of columns. The value of X(presently .33) is used as the cut-off percentage of columns which are considered "highly specified". In the future this value may be varied according to how specified a function is. It should be noted that the value of this variable(between 0 and 1) makes no difference when the function is fully specified or nearly fully specified. As mentioned in section 9.2, Cover_Set_Selection has no effect on the outcome of a decomposition when the function block to be decomposed is fully specified.
- *number_x_columns_in_each_partition()* is a subroutine which returns an array with a number assigned to each partition block(index of the array) equal to the number of highly specified columns in that partition block.
- *remove_repeated_columns()* is a subroutine which removes columns from all partitions except from the partition block which has the greatest number of highly specified columns.
- *load_output_data_structure()* is a subroutine which simply loads the output data structure. The form of the output is simply two arrays, one containing all the enumerated cubes(or minterms) in a specific order corresponding to the order of partition blocks which they are an element of. The other one having the number of elements in each partition block. The first array is named comp_row_list and the second array is named comp_size_list. These arrays are used by the dichotomy based input encoder.

9.4 Subroutines to convert data structures for *Cover_Set_Selection*

9.4.1 Function *convert_for_encoding()*

Name of Function	<i>:convert_for_encoding()</i>
Task	:To convert the format of the pig array and the PIb partition for encoding
Called where	<i>:gud_serial.c</i>
Input Arguments	<i>:bddmanager</i> Partition block PiB <i>Pig array</i> Number of bound partitions Number of cliques in pig array
Returns	<i>:NULL</i>
Functions called internally	<i>:Fill_min_cover()</i> .

9.4.2 Function *fill_min_cover()*

Name of Function	<i>:fill_min_cover()</i>
Task	:To convert the format the PIb partition for encoding
Called where	<i>:convert_for_encoding</i>
Input Arguments	<i>:PiB bdds</i>

row	x_0	x_1	x_2	x_3	x_4	y_0	y_1
0	0	0	0	0	0	0	0
1	0	0	0	1	1	0	1
2	0	0	0	1	0	1	-
3	0	1	1	0	0	0	1
4	0	1	1	0	1	0	0
5	0	1	1	1	0	-	1
6	0	1	0	0	0	0	0
7	1	1	0	0	0	0	-
8	1	1	1	1	0	0	0
9	1	1	1	0	0	1	0
10	1	1	1	1	1	0	1
11	1	1	1	1	0	-	1
12	1	0	0	0	1	0	0
13	1	0	1	1	1	-	-
14	1	0	1	1	0	1	0

Table 6: Truth table

Returns Min_Cover_Block
Functions called internally :Min_Cover_Block
 :bdd_compose().

10 ENCODING OF SYMBOLIC INPUTS BASED ON DYCHOTOMIES

10.1 COLUMN-COMPATIBLE CLASSES AS SYMBOLIC INPUTS TO H FUNCTION

The GUD algorithm constructs truth tables for the G and H functions, for a given bound set, at each stage of the decomposition. The minimal set of Maximal Compatible Classes (MCCs) is constructed using rough-set techniques and stored as an array of pointers to BDDs, one BDD per MCC, with non-zero terminal nodes indicating row indexes in the original truth table. An MCC can be interpreted as a symbolic input to the H successor function and as a symbolic output of the G predecessor function. By assigning unique binary code words to each MCC the complexity (calculated as the product and literal count) of both functions G and H is affected. The *code_gen* package in GUD implements constrained encoding of the MCCs as symbolic inputs to H . In the following, the various functions of *code_gen* subroutine will be explained.

10.2 CODE_GEN PARAMETERS

The input constraint matrix A is the main parameter passed to *code_gen* by *gud_main*. The matrix is constructed with the aid of following linked lists: (1) **out_onset_row_ptr**: list of row indexes in the original truth table for which the output(s) are non-zero (don't cares are allowed). (2) **comp_row_list**: list of row indexes in each MCC, starting with MCC_0 . (3) **comp_size_list**: list of MCC sizes (MCC size is the number of its row indexes.)

The lists are built in the *fill_pig* routine , which traverses the *Pig* array of BDDs. For example, given the two-output function :

with bound set = $\{x_0, x_1, x_4\}$ and free set = $\{x_2, x_3\}$.

$Pig = (14; 0, 2; 7, 8, 9, 11; 3, 5, 6; 12, 13; 1; 10; 4)$
 stored as bdds

row	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
1	0	0	0	0	0	1	0	0
2	0	1	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0
5	0	0	0	1	0	0	0	0
7	0	0	1	0	0	0	0	0
9	0	0	1	0	0	0	0	0
10	0	0	0	0	0	0	1	0
11	0	0	1	0	0	0	0	0
13	0	0	0	0	1	0	0	0
14	1	0	0	0	0	0	0	0

Table 7: Symbols of MCCs

and `out_onset_row_ptr =`

```
{14, 13, 11, 9, 5, 2; 13, 11, 10, 7, 5, 3, 2, 1} = {ON - sety0, ON - sety1}
comp_row_list = {14; 0, 2; 7, 8, 9, 11; 3, 5, 6; 12, 13; 1; 10; 4}
comp_size_list = {1; 2; 4; 3; 2; 1; 1; 1}
```

For the above function, A is given below, with $C_0 - C_7$ symbols for the respective MCCs:
 A is stored as file `con_file` and passed for further processing to `code_gen`.

10.3 SOLVING THE INPUT CONSTRAINTS WITH THE SIS DICHOTOMY SOLVER PACKAGE

The constraints are first read using the file i/o `read_cons()` in the standard distribution `/sis/enc` package.
Reduced seed dichotomies are generated next with

```
red_list = reduce_seeds(gen_uniq(list))
```

All prime dichotomies are then generated with

```
gen_eqn(red_list, LIMIT)
```

and a minimal set of primes which cover all seeds is selected with:

```
table = dic_to_sm(prime.list, red_list);
cover = sm_minimum_cover(table, (int*)0, 0, 0);
```

For the above constraints, the cover consists of three dichotomies :

```
00110101 ; 11001010
01101001 ; 10010110
00111010 ; 11000101
```

where the left side of each dichotomy is the ones side

After the `mccbinary` codewords are extracted from the left side of each final dichotomy, they are converted to the format of the `pigbinary` array, a parameter to the `fill_table()` routine for generating the G and H tables.

The eight code words for the above example are then:

```
 $C_0 = 000, C_1 = 010, C_2 = 111, C_3 = 101, C_4 = 011$ 
 $C_5 = 100, C_6 = 001, C_7 = 110$ 
```

11 ENCODING MINTERMS/CUBES OF THE BOUND SET VARIABLES

The current version of `code_gen()` encodes fully specified products of the bound set variables, i.e. their minterms. This allows for easy verification of G and H tables versus original table with the `/sis/verify` package. The dichotomy-based algorithm can be extended to encode cubes of the bound set variables as well and this is one of our areas for further research.

12 PRINTING G AND H TABLES

Name of subroutine	: <i>print.c</i>
Task	:To print the PIg and PIh tables
Input	:An array containing the maximum cliques (The PIg array) (shown in Figure 8)
Output	:The reduced PIg and PIh tables

The *print.c* subroutine is a part of GUD that works together with subroutines from section 8. *treader.c* is the reader that inputs the input function and creates BDDs. Programs that create partitions for the free variables, bound variables and the output variables, form the maximum cliques by solving the clique covering and create the π_g BDDs's are presented in section 8. *print.c* takes the π_g bdd's and prints the *G* and *H* tables. Thus this completes one step of the decomposition, as the next step the above processes have to be repeated until the function is decomposed to the assumed type of blocks. The various functions used in *print.c* will be explained below.

12.1 SUBROUTINES TO FIND FREE AND BOUND SETS OF VARIABLES

Name of subroutine	: <i>Travbound1.c</i>
Task	:To find number of elements in bound/free sets.
Called where	: <i>main.c</i>
Input Arguments	:Link list of bound/free BDD pointers.
Returns	:Integer giving the number of elements in bound/free set.
Functions called internally	:None.

Name of subroutine	: <i>Travbound2.c</i>
Task	:To find which elements are in bound/free sets.
Called Where	: <i>main.c</i>
Input Arguments	:Link list of bound/free BDD pointers.
Returns	:Array containing variable id of elements in bound/free set.
Functions called internally	:None.

Partition_block Link List Data Structure is shown in Figure 7.

```
typedef struct partition_block
{
    int      block_id;
    bdd      bdd_root;
    struct partition_block *next;
};
```

block_id :0 for zero BDD
block_id :1 for one BDD
block_id :2 for two BDD and so on
bdd_root :pointer to head of BDD.

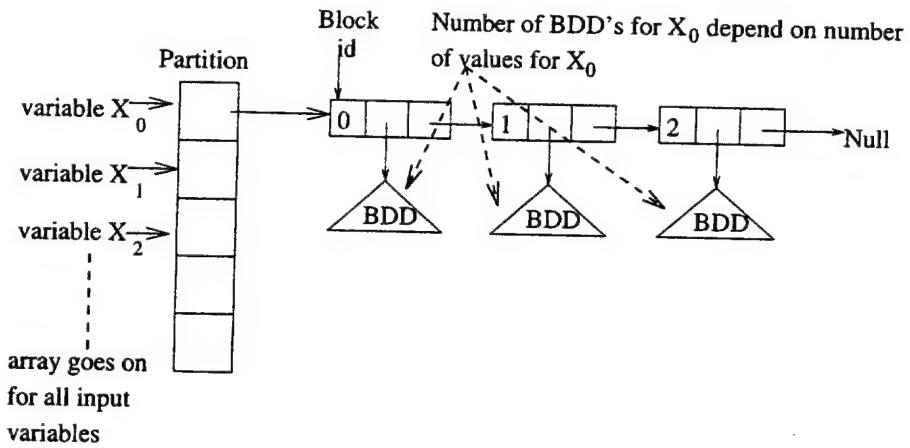


Figure 7: Partition-block Link List Data Structure

12.1.1 Some Definitions

b	: Pointer to bound set link list
Location	: <i>gud.h</i>
a	: Pointer to free set link list
Location	: <i>gud.h</i>
bdd_zero	: bdd terminating in node zero;
bdd_one	: bdd terminating in node one;
0 BDD	: BDD formed by 0's in a column of input table;
1 BDD	: BDD formed by 1's in a column of input table;

In the *gud.h* there is a link list as shown in Figure 7 which points to the BDD's of the input and output variables. This link list has a structure such that it has a pointer to the 0 BDD of the first variable and a pointer to the next element in the link list, which in turn has a pointer to the 1 BDD of the first variable in the input and a pointer to null if the input function is two valued. If the function is multivalued then this link list goes on for the number of values. And there is an array of such link list's where each element of the link list corresponds to a particular input variable. The structure of this link list is as shown above in Figure 7. There exist two functions in *print.c* which are used to traverse the bound (*b*) and free(*a*) link list for the purpose of finding out which elements are in bound set and which elements are in the free set, respectively.

Travbound1 traverses the bound/free set link list and increments an integer(*i* in *Travbound1.c*) for every element in the bound or free set. This integer informs how many variables are in the bound or free set.

Travbound2 traverses the link list and stores in an array which of the input elements are in the bound or free set, respectively.

2.1.2 Subroutine Print_partition

Name of subroutine	:Print_Partition
sk	:To traverse the input BDD's

Lled where :*fill_table*

Input Arguments	:Pointer to head of BDD link list, code indicating path to traverse.
Returns	:Integer(1,0 or dont care)
1	:path traversed is in one BDD
0	:path traversed is in zero BDD
dont care	:path traversed is in both BDD's

Subroutines called internally: 1) *bdd_compose*. Function:Traverses BDD along specific path and returns a BDD; 2) *no_of_vars*. Function:Find out number of variables in the BDD.

Print_Partition is passed two parameters: (1) A pointer, which points to the head of the link list of the BDD's. (2) A code which tells the function which path to go along the BDD.

Since the input table will be too large to fit in memory this function is used to traverse each of the BDDs of the input variables, along specific paths so that the π_g (PIg) and π_h (PIh) tables can be printed, one row at a time.

Subroutine *Print_Partition* first calls subroutine *no_of_vars* which is passed a pointer to the BDD to be traversed and it returns the number of variables in the BDD. Then *Print_Partition* calls *bdd_compose* which is given a path along the BDD (for example 000) and then *bdd_compose* will be called three times for this example and will return a BDD which will either point to terminating nodes 1 or 0, depending on whether this particular path is present in the BDD or not, respectively.

Example 11.1 explaining *print_partition*.

This example will illustrate how subroutine *print_partition* works on an example.

bdd_zero will be a BDD terminating in node zero, *bdd_one*, a BDD terminating in node one, 0 BDD is a BDD formed by 0's in column of input table, 1 BDD is a BDD formed by 1's in column of input table.

Let us assume path = 0 0 1, and let original BDD = *bddo*. Let returned BDD = *bddr,bddr1*; The following steps will illustrate the operation of the algorithm.

```

step1:Let bddo = 0 BDD
step2:bddr = bdd_compose(bddo, 0)
step3:bddr1= bdd_compose(bddr, 0) /*Now pass bdd_compose the reduced bdd*/
step4:bddr = bdd_compose(bddr1,1)
step5:Store bddr
step6:Let bddo = 1 BDD
step7:Repeat Steps 2-5
step8:If both bddr == 1 then return dont care (indicates that path is present
      in both BDD's)
      :If only bddr using 0 BDD == 1 return 0
      :If only bddr using 1 BDD == 1 return 1

```

2 SUBROUTINE FILL_PIG

ne of subroutine	<i>Fill_Pig</i>
k	:To traverse the PIg array (Figure 8) out which rows in the input table are compatible
ed where	: <i>main.c</i>
it Arguments	:Pointer to head of PIg array, and a symbolic code.
irns	:Pigcode Array (Figure 9)

Subroutines called internally 1) *bdd_compose*. Function:Traverses BDD along specific path and returns a BDD; 2) *no_of_vars*. Function:Find out number of variables in the BDD.

The PIg array is formed in the following three steps:

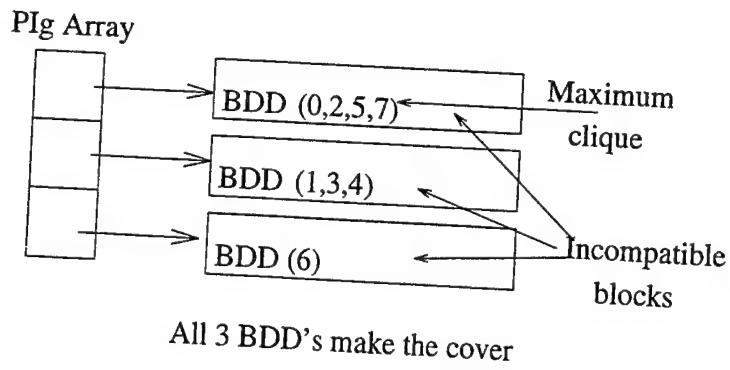


Figure 8: PIg Array

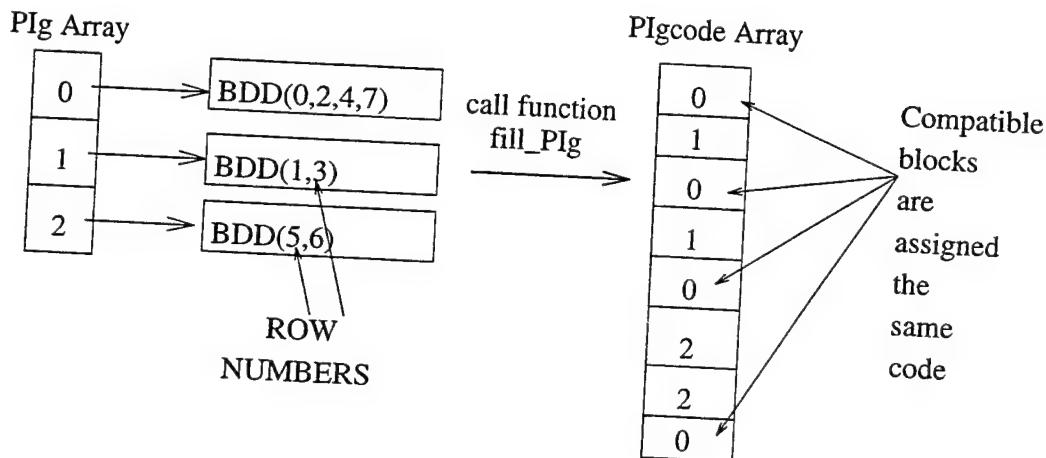


Figure 9: PIg Array and Pigcode Array

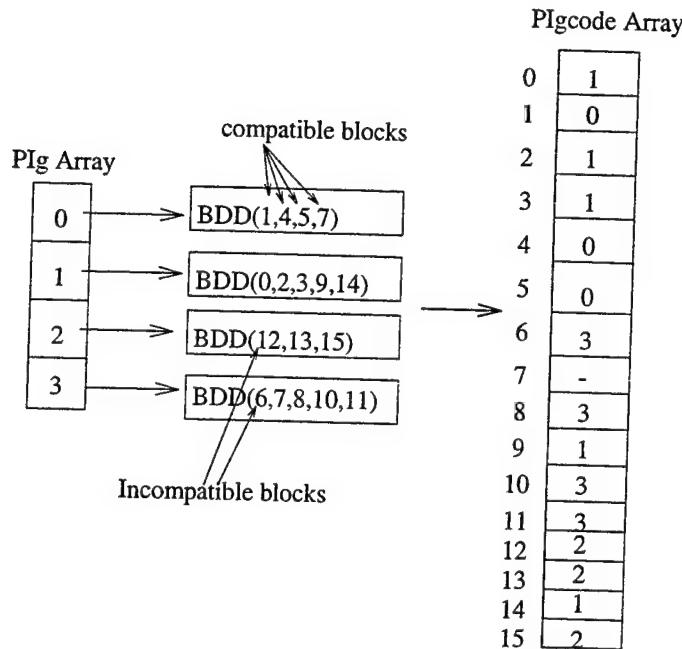


Figure 10: The PIg and PIgcode Arrays for Example 11.2

- Form pairwise compatible classes.
- Find maximum cliques, the algorithm used is a greedy algorithm.
- Find the cover.

The *PIg* array is shown in Figure 8.

12.2.1 Example 11.2

Figure 10 shows the *PIg* array with four elements. The first element is pointing to a BDD made of 1,4,5,7, where the numbers correspond to row numbers. The second element is pointing to 0,2,3,9,14. Hence this means that rows 1,4,5 must be assigned one code (for example 0) and rows 0,2,3,9,14 must be assigned a different code (for example 1) telling us that these two blocks are incompatible and since 7 is in the first and the fourth block it can be assigned a dont care. The *fill_pig* function goes through each element in this *PIg* array and finds out which rows are compatible and which are not, and then the *fill_pig* function assigns codes to these rows. In the future some encoding scheme will be incorporated into the program.

The *pigcode* array is used to store the symbolic codes. *Fill_Pig* uses the *bdd_compose* subroutine and goes along a specific path (example 001) of the BDD of the first element of the *PIg* array and if this path exists in the BDD then it assigns a code (example 0) to element 1 (path 001), thus it goes along all possible paths of the first BDD and stores 0 in all corresponding elements of the *pigcode* array. Similarly it goes through all the elements of the *PIg* array and fills up the *pigcode* array such that each element of the array corresponds to the code of that particular row.

12.3 SUBROUTINE FILL_TABLE

Name of Subroutine :*Fill_table*

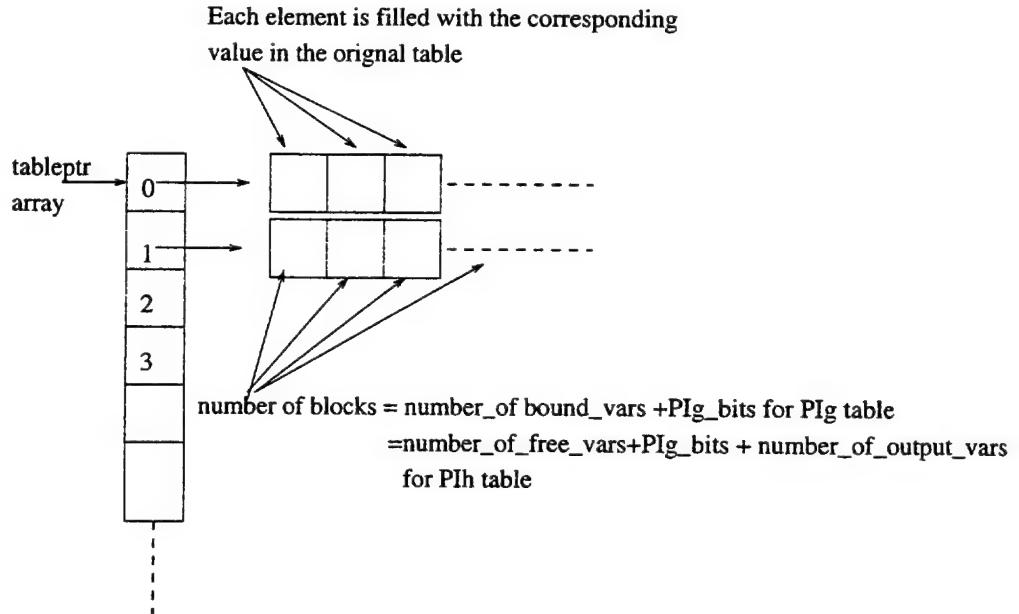


Figure 11: Tableptr Array

Task	:To fill up an array with the <i>PIg</i> or <i>PIh</i> tables
Called where	: <i>main.c</i>
Input Arguments	:Number of bound free variables, array telling which elements are in bound or free sets, <i>Pigcode</i> array
Returns	:Pointer to an array of pointers to the rows of the reduced <i>PIg</i> or <i>PIh</i> table

Example 11.3 of operation of fill_table

Since the input table will be too large to fit in memory the *fill_table* subroutine is used to print the *PIg* and *PIh* tables row-by-row and delete rows which are redundant, i.e. if they are included in any other row. So as a first step this subroutine calls *print_partition* by passing it the code for the first row (example 000) and a pointer to the first variable in the input table and the print partition returns either a 0 a 1 or a dont care which tells us that that in row 0 for variable 0 there is a 0, 1 or don't care. This information is stored in an array called *tableptr* (shown in Figure 11) which is being dynamically created. This process is repeated for each variable in the row (bound set variables for table *PIg* and free set variables for table *PIh*). Once an entire row has been stored in the *tableptr* array the *PIg* code of that particular row is filled in the *tableptr* array. This code is obtained from the *pigbinary* array which is obtained by converting the integer codes in the *pigcode* array into binary codes and storing them in the *pigbinary* array which is shown in Figure 12. This is done by a subroutine called *int_to_bin*. Now each element of this row is compared with each corresponding element of any other rows which may have been stored in the *tableptr* array. If the new row is found to be included in any of the previous rows then the row with the most number of dont cares is kept and the other redundant row with less dont cares is deleted and memory is freed. Thus this process is repeated for all the rows and the *pigcode* array is filled with the *PIg* table or *PIh* table. Finally as the last step the printing of the table is done by a subroutine called *print_table* which prints the table in BLIF format.

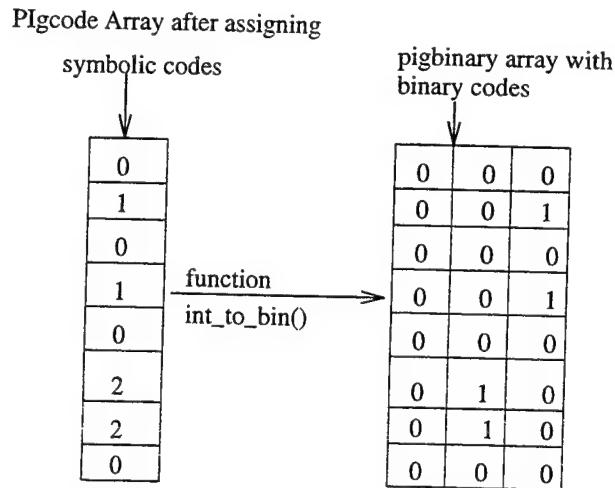


Figure 12: PIgbinary Array

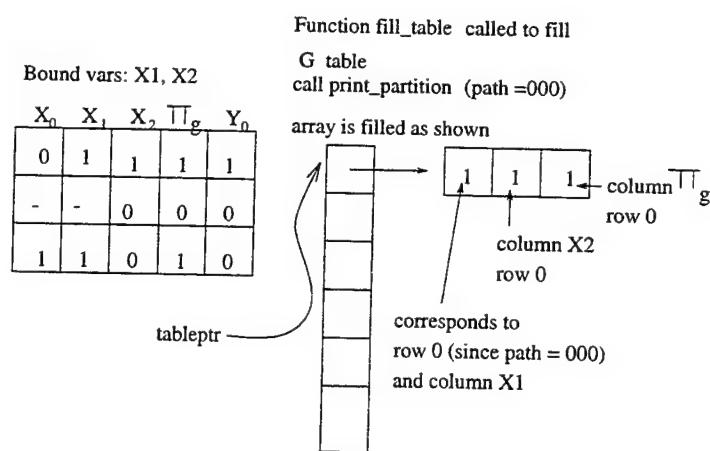


Figure 13: Example of operation of subroutine `fill_table`

12.4 Function *out_esp()*

Name of Function	<code>:out_esp()</code>
Task	:To print Espresso files.
Called where	<code>:gud_serial.c</code>
Input Arguments	<code>:Number of encoding bits.</code> <code>:Number of bound variables</code> <code>:Number of free variables</code> <code>:Number of output variables</code> <code>:Array indicating which are the bound/free variables</code> <code>:tableptr array containing the G/H table</code> <code>:Espresso file to write to</code>
Returns	<code>:NULL</code>
Functions called internally	<code>:print_esp().</code>

12.5 Function *print_esp()*

Name of Function	<code>:print_esp()</code>
Task	:To print Espresso files.
Called where	<code>:out_esp.c</code>
Input Arguments	<code>:new_no:Number of bound variables if G block or</code> <code>number of free variables + number of encoding bits if H block</code> <code>:new_no1:Number of encoding bits if G block</code> <code>or Number of output variables if H block</code> <code>:tableptr array containing the G/H table</code> <code>:Espresso file to write to</code>
Returns	<code>:NULL</code>
Functions called internally	<code>:NONE.</code>

12.6 Functions to print blif file of decomposed function

12.7 function *writefile()*

Name of Function	<code>:writefile()</code>
Task	:To print blif file.
Called where	<code>:gud_serial.c</code>
Input Arguments	<code>:Number of input partitions.</code> <code>:Number of output partitions</code> <code>:Number of bits for encoding</code> <code>:bf_no:Number of bound variables for G</code> <code>:Number of free variables for H</code> <code>:Number of output variables</code> <code>:Array indicating which are the bound/free variables</code> <code>:tableptr array containing the G/H table</code>
Returns	<code>:NULL</code>
Functions called internally	<code>:print_blif().</code>

12.7.1 function *print_blif()*

Name of Function	<code>:print_blif()</code>
-------------------------	----------------------------

Task	:To print blif file.
Called where	: <i>writefile()</i>
Input Arguments	:tableptr array containing the G/H table new_no:Bound number for G or free number + encoding bits for H new_no1:Bound number for G or free number + encoding bits for H
Returns	:NULL
Functions called internally	:None.

12.8 PRINTING OUTPUT IN BLIF FORMAT

```
.model example
.inputs a b c d e
.outputs f
.names c d e x
1-1 1
010 1
.names c d e y
0-- 1
--1 1
-1- 1
.names a b x y f
0-0- 1
-1-1 1
1-1- 1
.end
```

Blif Format is a multi-level description of the Boolean network. Each node in this representation has a single output. Therefore, each net (or signal) has only a single driver, and one can therefore name either the signal or the gate which drives the signal without ambiguity.

.model example: specifies the name of the model (example).
 .inputs a b c d e: gives the name of the input variables (a, b, c, d, e).
 .outputs f: gives the name of the output of the function (f).
 .names a b c x: with the following ON set describes the logic of a node (sub-block M) in Figure 14.
 The input variables to this node are a, b, c and the output variable is y.

.names d e x y f: with the following ON set describes the logic of a node (sub-block O) in Figure 14.
 The input variables to this node are d, e, x, y, and the output variable is f.

.end: marks the end of the model.

12.8.1 Example 11.4

Let the input table be as shown below

```
.i 4
.o 1
.a X1 X3
```

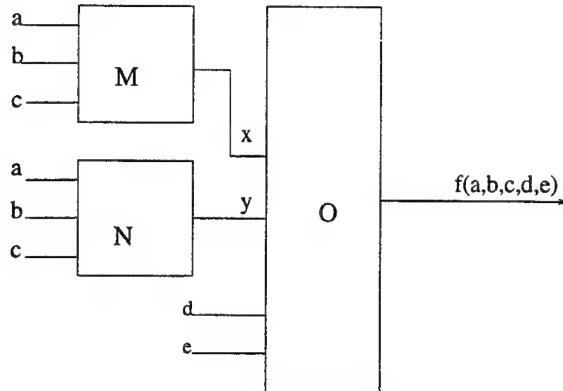


Figure 14: Explanation of BLIF Format

	X0	X2	
X1	X3	\	00 01 11 10
	00	0	0 0
	01	1	1 1
	11	0	1 1
	10	1	1 1

Figure 15: Karnaugh Map for Decomposition of function from Example 11.4

```
.b X0 X2
-0-0 0
01-1 0
1--1 1
11-- 1
00-1 1
-100 1
-110 1
.end
```

This corresponds to the Karnaugh map from Figure 15 in which columns correspond to bound variables and rows to free variables.

G Table

In order to print the G table we need the table of F , bound set variables and the PIg column. First subroutine *travbound1* will return 2 which is the number of variables in the bound set. Next subroutine *travbound2* will be called and it will return an array in which the first element will have 0 and the second will be 2, which are the variables in the bound set, i.e. X_0 and X_2 in the example. Now function *fill_pig* will be called and it will fill up the *pigcode* array with the symbolic codes for the rows. Then function *Fill_table* will be called which will call function *print_partition* for the first row and print partition will fill up the first row of the *pigcode* array with don't care (-) which corresponds to row 0, variables 0 and 2 (bound set variables) and the PIg code which in this case is dont care (-). Since this row has all dont cares it is eliminated. Then the next row is filled in the *tableptr* array which in this case is 0 - 1 where the last bit (1) is the code for the PIg column. This process is repeated with all rows and if any row is found to be identical to any previous row it is eliminated. The resultant unreduced and reduced tables

are shown below.

Unreduced G Table

```
.model example
.inputs X0 X1 X2 X3
.outputs Y0
.names X0 X2 g0
-
0 - 1
1 - 0
1 - 0
0 - 1
- 0 -
- 1 -
```

In the above tables column 1 corresponds to variable X_0 and the second column corresponds to variable X_2 and the third column is the PIg column. Now as can be seen from the above table one bit is required for encoding PIg telling us that column multiplicity was 2 and there will be one output from the G block. Figure 16 shows how the decomposition will look like. The codes for the PIg column are arbitrarily chosen. A dont care in the third PIg column means that that row code occurs in more than one clique. Next the reduced G table is printed as shown below.

Reduced G table

```
.model example
.inputs X0 X1 X3 X4
.outputs Y0
.names X0 X2 g0
0 - 1
1 - 0
```

H Table

The H table is printed in the same way except the H table consists of the free variables, the PIg column and one output variable at a time, hence for a multiple output function the table shown below would have been repeated for each output variable

The unreduced H table is as shown in table below, and Figure 16 presents the decomposed structure with subfunctions $g_0(X_0, X_2)$ and $H(g_0, X_1, X_3)$.

Unreduced H Table

```
.model example
.inputs X0 X1 X3 X4
.outputs Y0
.names X1 X3 g0 Y0
0 0 - 0
1 1 1 0
- 1 0 1
1 - 0 1
0 1 1 1
1 0 - 1
1 0 - 1
```

Reduced PIh table

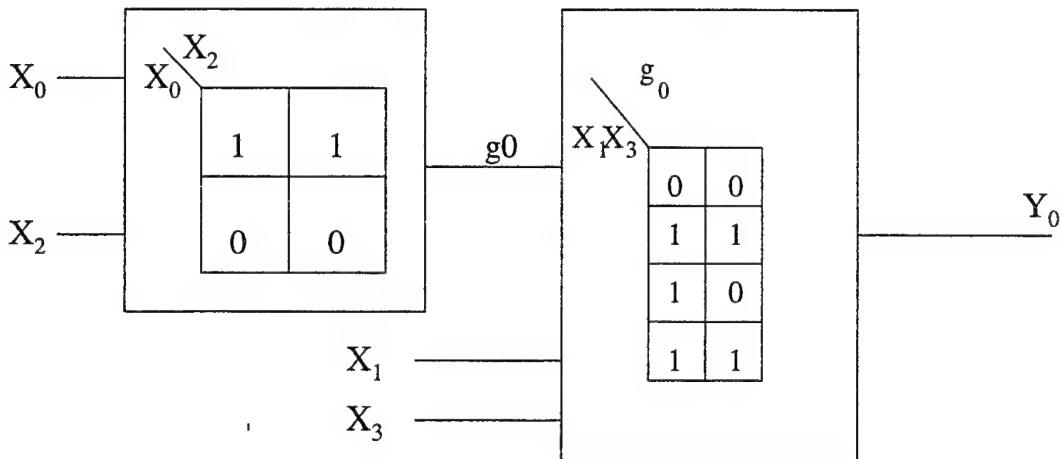


Figure 16: Decomposed Function

```
.model example
.inputs X0 X1 X3 X4
.outputs Y0
.names X1 X3 g0 Y0
0 0 - 0
1 1 1 0
- 1 0 1
1 - 0 1
0 1 1 1
1 0 - 1
```

13 Results

Categories:

- A: Total number times when DFC was the lowest(not a tie)
- B: Total number times when the lowest DFC was a tie
- C: percentage of tests when DFC was the lowest(i.e. $(A+B)/(total \# functions) * 100$)
- D: Average execution time per function in seconds
- E: Longest execution time in seconds
- F: Accumulative DFC
- G: Average DFC

Strategy codes:

The number "1" designates that the strategy used always started by trying Ashenhurst(single output) decompositions first. If none exist or the limit has been reached(100 partitions have been checked) then Curtis like decompositions are searched for next.

A "z" simply means that only one algorithm is available for partitioning, column compatibility, set covering and encoding(i.e. there are no options to select different algorithms in DEMAIN).

Table 8: COMPARISON OF DEMAIN AND VARIATIONS OF GUD

Program				DEMAIN		GUD			
name	in	out	cubes	Strategy 1-z		Strategy 1-a		Strategy 1-b	
				DFC	time(s)	DFC	time(s)	DFC	time(s)
psu_add0_70	8	1	77	28	1.1	64	209.3	28	9.1
psu_and_or_chain8_70	8	1	77	28	1.2	24	23.6	32	21.6
psu_ch176f0_70	8	1	77	28	0.9	24	3.8	24	3.9
psu_ch177f0_70	8	1	77	20	0.7	20	3.3	20	3.9
psu_ch22f0_70	8	1	77	52	1.4	40	30.1	44	10.2
psu_ch30f0_70	8	1	77	40	1.3	44	43.3	44	15.5
psu_ch47f0_70	8	1	77	56	2.7	64	140.6	44	48.6
psu_ch52f4_70	8	1	77	68	2.3	64	172.1	68	456.9
psu_ch70f3_70	8	1	77	68	2.6	52	386.8	28	7.1
psu_ch74f1_70	8	1	77	64	2.5	76	227.0	40	42.5
psu_check_fail_70	8	1	77	56	2.2	56	170.4	60	57.4
psu_contains_4_ones_70	8	1	77	80	24.4	76	688.1	76	2118.2
psu_greater_than_70	8	1	77	28	1.6	44	954.5	44	19.7
psu_interval1_70	8	1	77	56	2.5	52	325.2	56	108.0
psu_interval2_70	8	1	77	124	2.6	76	785.3	120	2814.3
psu_kdd1_70	8	1	77	28	1.1	16	4.5	16	6.2
psu_kdd2_70	8	1	77	28	0.8	28	3.7	24	4.7
psu_kdd3_70	8	1	77	28	1.2	28	23.3	24	5.7
psu_kdd4_70	8	1	77	12	0.7	28	10.3	12	3.9
psu_kdd5_70	8	1	77	44	1.7	40	15.9	40	11.4
psu_kdd6_70	8	1	77	28	1.0	28	3.9	28	5.1
psu_kdd7_70	8	1	77	64	2.3	28	3.9	28	4.7
psu_kdd8_70	8	1	77	28	1.0	24	20.2	24	5.6
psu_kdd9_70	8	1	77	28	0.9	28	25.5	28	5.4
psu_kdd10_70	8	1	77	28	0.9	28	23.9	24	5.8
psu_majority_gate_70	8	1	77	60	2.7	56	220.3	56	68.3
psu_modulus2_70	8	1	77	60	2.2	72	182.2	56	423.2
psu_monkish1_70	8	1	77	28	1.0	28	7.8	24	4.8
psu_monkish2_70	8	1	77	40	1.9	40	71.3	40	47.3
psu_monkish3_70	8	1	77	40	1.4	28	10.9	28	5.1
psu_mux8_70	8	1	77	28	1.8	52	54.7	48	128.2
psu_nnr1_70	8	1	77	40	2.5	40	48.5	40	55.1
psu_nnr2_70	8	1	77	40	1.4	28	31.8	32	10.5
psu_nnr3_70	8	1	77	68	1.8	68	716.2	68	460.4

Table 9: COMPARISON OF DEMAIN AND VARIATIONS OF GUD

Program				DEMAIN		GUD			
name	in	out	cubes	Strategy 1-z		Strategy 1-a		Strategy 1-b	
				DFC	time(s)	DFC	time(s)	DFC	time(s)
psu_or_and_chain8_70	8	1	77	32	0.9	28	4.5	28	4.8
psu_pal_70	8	1	77	28	0.9	28	4.1	28	3.8
psu_pal_dbl_output_70	8	1	77	132	2.2	104	371.6	92	522.6
psu_parity_70	8	1	77	28	0.9	28	3.8	28	3.5
psu_primes8_70	8	1	77	92	20.3	80	741.5	56	367.5
psu_remainder2_70	8	1	77	84	1.9	80	164.2	60	372.5
psu_rnd1_70	8	1	77	140	31.3	140	1447.5	176	2789.4
psu_rnd2_70	8	1	77	148	12.4	132	1306.6	172	3565.0
psu_rnd3_70	8	1	77	144	6.4	112	960.5	160	2702.3
psu_rnd_m10_70	8	1	77	28	1.7	28	8.4	28	5.5
psu_rnd_m1_70	8	1	77	28	1.3	28	3.4	28	3.7
psu_rnd_m25_70	8	1	77	64	4.6	64	187.4	68	393.0
psu_rnd_m50_70	8	1	77	100	23.1	92	661.8	116	2453.5
psu_rnd_m5_70	8	1	77	28	1.6	28	8.1	28	4.7
psu_rndvv36_70	8	1	77	88	3.5	88	257.1	92	80.4
psu_substr1_70	8	1	77	92	6.9	92	292.4	92	461.4
psu_substr2_70	8	1	77	144	6.0	80	232.4	92	2363.7
psu_subtraction1_70	8	1	77	128	9.7	128	392.8	140	723.3
psu_subtraction3_70	8	1	77	12	0.8	20	3.6	20	3.6

Table 10: Summary of Results for tables 13 and 12 for FLASH functions with 70% don't cares

Program	Strategy	Category					
		A	B	C	D(sec)	E(sec)	F
DEMAIN	1-z	4	19	43%	6	31.3	3056
GUD	1-a	10	26	68%	220	1447.0	2844
GUD	1-b	11	23	64%	420	3565.0	2872

An "a" signifies the following options selected for the program GUD:

- 1) GUD partitioning method 1
- 2) column compatibility method 1 (set covering)
- 3) encoding method 3 (dichotomy encoding w/ set cover selection)

A "b" signifies the following options selected for the program GUD:

- 1) GUD partitioning method 2 (all bound sets of specified size)
- 2) column compatibility method 1 (set covering)
- 3) encoding method 3 (dichotomy encoding w/ set cover selection)

For more details on algorithms used see the respective sections in this document or in the report.

Categories:

A: Number of Partitions tried

B: Decomposition was found using one of the partitions selected?

For more details on algorithms used see the respective sections in this document or in the report.

References

- [1] E.M.Sentovich, K.J. Singh, A. Saldanha, R. Brayton, A. S. Vincentelli "SIS : A System for Sequential Circuit Synthesis", UCB ERL Memorandum No. M92/41
- [2] R.Murgai, R.Brayton, A.S. Vincentelli "Optimum Functional Decomposition Using Encoding", 31st DAC 1994, pp.408-414, ACM/IEEE
- [3] SIS 1.2 release notes
- [4] T. Luba, M.A. Markowski, B. Zbierzchowski, "Logic Decomposition for Programmable Gate Arrays," *Proc. of Euro-ASIC'92*, IEEE Computer Society Press, pp. 19-24, Paris 1992.
- [5] T. Luba, R. Lasocki, J. Rybnik, "An Implementation of Decomposition Algorithm and its Application in Information Systems Analysis and Logic Synthesis," *International Workshop on Rough Sets and Knowledge Discovery*, pp. 487-498, Banff 1993.
- [6] T. Luba, M. Mochocki, J. Rybnik, "Decomposition of Information Systems Using Decision Tables," *Bulletin of the Polish Academy of Sciences, Technical Sciences*, Vol. 41, N o.3, 1993.
- [7] T. Luba, R. Lasocki "Decomposition of Multiple-Valued Boolean Functions", *Appl. Math. and Comp. Sci.*, 1994, vol. 4, No. 1, 125-138.
- [8] T. Luba Multiple-Valued and Multi-Level Decomposition and its Applications in Information Systems Analysis and Logic Synthesis, 1995. (Where ???)
- [9] M. A. Perkowski, T. Luba, S. Grygiel, et al Generalized Universal Decomposer, main report, 1995.
- [10] M. A. Perkowski, M. Burns, et al Encoding for Functional Decomposition, report, 1995.
- [11] M. A. Perkowski, M. Burns, et al Search strategies for Functional Decomposition, report, 1995.
- [12] Swamy, M.N.S., K. Thulasiraman, Graphs, Networks, and Algorithms, John Wiley Sons, New York, 1981, p. 250.

Table 11: COMPARISON OF DEMAIN AND VARIATIONS OF GUD

Program				GUD		DEMAIN	
				Strategy 1-b	Strategy 1-z	DFC	time(s)
Benchmarks	in	out	cubes	DFC	time(s)	DFC	time(s)
psu_add0	8	1	256	28	53	28	12
psu_add2	8	1	256	28	15	28	4
psu_add4	8	1	256	20	14	20	3
psu_and_or_chain8	8	1	256	28	50	28	14
psu_ch15f0	8	1	256	84	107	80	11
psu_ch176f0	8	1	256	24	18	28	4
psu_ch177f0	8	1	256	20	15	20	3
psu_ch22f0	8	1	256	28	21	28	4
psu_ch30f0	8	1	256	44	62	52	7
psu_ch47f0	8	1	256	76	92	80	10
psu_ch52f4	8	1	256	224	14107	220	331
psu_ch70f3	8	1	256	56	557	56	21
psu_ch74f1	8	1	256	112	6239	112	197
psu_ch83f2	8	1	256	168	213		
psu_ch8f0	8	1	256	52	85	44	7
psu_contains_4_ones.s	8	1	256	76	4511	76	220
psu_greater_than	8	1	256	28	40	28	12
psu_interval1	8	1	256	152	7838	136	317
psu_interval2	8	1	256	120	368		
psu_kdd1	8	1	256	16	10	28	4
psu_kdd2	8	1	256	24	11	28	4
psu_kdd3	8	1	256	24	12	28	5
psu_kdd4	8	1	256	12	8	20	4
psu_kdd5	8	1	256	64	434	68	21
psu_kdd6	8	1	256	28	12	28	4
psu_kdd7	8	1	256	28	11	28	5
psu_kdd8	8	1	256	24	11	28	5
psu_kdd9	8	1	256	28	12	28	5
psu_kdd10	8	1	256	24	14	28	5
psu_majority_gate	8	1	256	76	4807	80	216
psu_modulus2	8	1	256	76	517	68	21
psu_monkish1	8	1	256	24	10	28	4
psu_monkish2	8	1	256	56	60	56	11
psu_monkish3	8	1	256	32	9	32	4

Table 12: COMPARISON OF DEMAIN AND GUD ON FULLY SPECIFIED FLASH BENCHMARKS

Program				GUD		DEMAIN	
Benchmarks	in	out	cubes	Strategy 1-b	Strategy 1-z	DFC	time(s)
psu_mux8	8	1	256	52	55	32	5
psu_nnr1	8	1	256	36	338	36	25
psu_nnr2	8	1	256	32	17	32	5
psu_nnr3	8	1	256	240	8614	244	365
psu_or_and_chain8	8	1	256	28	64	28	13
psu_pal	8	1	256	28	18	28	7
psu_pal_dbl_output	8	1	256	188	679	180	50
psu_pal_output	8	1	256	368	5528	372	431
psu_parity	8	1	256	28	11	28	4
psu_primes8	8	1	256	368	7024	212	326
psu_remainder2	8	1	256	296	7070	180	316
psu_rnd1	8	1	256	376	438		
psu_rnd2	8	1	256	332	6713	352	902
psu_rnd3	8	1	256	360	5824	364	926
psu_rnd_m1	8	1	256	28	12	28	5
psu_rnd_m10	8	1	256	108	6708	108	185
psu_rnd_m25	8	1	256	272	301		
psu_rnd_m5	8	1	256	80	6450	80	137
psu_rnd_m50	8	1	256	344	6233	336	778
psu_rndvv36	8	1	256	92	86	92	14
psu_substr1	8	1	256	80	3839	80	536
psu_substr2	8	1	256	112	3916	92	512
psu_subtraction3	8	1	256	20	10	20	3

Table 13: COMPARISON OF PARTITIONING APPROACHES

Partitioning Method				Method #0			DEMAIN			GUD(b)		
Benchmarks	in	out	cubes	A	B	time(s)	A	B	time(s)	A	B	time(s)
psu_add0_70	8	1	77	1	y	0.2	2	y	0.4	5	y	2.9
psu_and_or_chain8_70	8	1	77	1	y	0.3	4	y	0.2	5	y	2.9
psu_ch176f0_70	8	1	77	1	y	0.3	1	y	0.3	1	y	1.1
psu_ch177f0_70	8	1	77	1	y	0.2	1	y	0.1	1	y	0.1
psu_ch22f0_70	8	1	77	2	y	0.2	1	y	0.2	1	y	1.1
psu_ch30f0_70	8	1	77	3	y	0.4	2	y	0.3	6	y	3.3
psu_ch47f0_70	8	1	77	4	y	0.3	2	y	0.2	4	y	2.2
psu_ch52f4_70	8	1	77	4	n	0.3	1	y	0.3	1	y	1.0
psu_ch70f3_70	8	1	77	1	y	0.2	2	y	0.2	3	y	1.9
psu_ch74f1_70	8	1	77	1	y	0.4	2	y	0.3	2	y	1.5
psu_check_fail_70	8	1	77	3	y	0.4	1	y	0.2	4	y	2.5
psu_contains_4_ones_70	8	1	77	none	exist							
psu_greater_than_70	8	1	77	3	n	0.3	16	y	0.5	13	y	6.8
psu_interval1_70	8	1	77	3	n	0.3	5	y	0.3	15	y	8.0
psu_interval2_70	8	1	77	none	exist							

[13] Wei Wan, "A New Approach to the Decomposition of Incompletely Specified Functions Based on Graph Coloring and Local Transformation and Its Application to FPGA Mapping," Master's Thesis, Portland State University, 1992.

Superresolution of Passive Millimeter-Wave Imaging

Stanley J. Reeves
Associate Professor
Department of Electrical Engineering

Auburn University
200 Broun Hall
Auburn, AL 36849

Final Report for:
Summer Research Extension Program

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base
Washington, DC

and

Wright Laboratory

December 1995

Superresolution of Passive Millimeter-Wave Imaging

Stanley J. Reeves
Associate Professor
Department of Electrical Engineering
Auburn University

Abstract

Passive millimeter-wave (PMMW) imagery has tremendous potential for imaging in adverse conditions; however, poor resolution poses a serious limitation to this potential. Experimental results indicate that image processing alone cannot significantly superresolve general extended targets beyond the measured spatial frequency region. Furthermore, the potential benefits of various acquisition strategies is quite limited. A combination of acquisition strategies and image processing techniques holds the best promise for superresolution. We propose a novel acquisition method that allows one to increase the measured spatial frequency bandwidth of the data beyond the diffraction limit. Along with this, we propose to develop a new image restoration algorithm that incorporates all the measured data into a single restored higher-resolution image. This will lay the foundation for previously unachievable superresolution.

Superresolution of Passive Millimeter-Wave Imaging

Stanley J. Reeves

1 Introduction

Objects generally radiate across the entire frequency spectrum of electromagnetic energy. The radiation at visual and infrared wavelengths can be measured across a range of angles to form images. Images can also be acquired at millimeter wavelengths. Passive millimeter-wave (PMMW) images are particularly useful for navigation, guidance, and surveillance because millimeter waves penetrate fog and heavy rain and radiate at night as well as during daylight. Furthermore, millimeter waves penetrate smoke and small debris, making PMMW imagery especially advantageous under battlefield conditions [1]. Furthermore, the passive nature of this imaging modality makes stealth possible and reduces the opportunity for countermeasures. PMMW imaging also has a number of potential civilian applications, including aircraft landing, collision avoidance, and airport security screening.

A major drawback to PMMW images is poor angular resolution. Resolution is inversely proportional to wavelength and proportional to the size of the aperture. Because PMMW radiation has a comparatively long wavelength relative to visual radiation, the resolution is much lower. Furthermore, the use of very large apertures is not usually an option because of platform size constraints. For example, an aperture may have to fit within the diameter of a guided bomb or missile. To make PMMW useful for bomb guidance, the spatial resolution must be improved many times. In other applications, the necessary resolution improvement may not be as demanding, but higher resolution is still required to make these applications feasible.

Resolution improvement must be pursued via two strategies: image acquisition and image processing. It is our view that neither strategy is capable of solving the problem alone. The potential improvement of each strategy in isolation is quite limited, and some approaches may require both novel acquisition techniques and appropriate image processing of the resulting data to achieve their goal. Thus, this project considers the two strategies in an integrated way. In the next section, we define the problem and discuss the assumptions required to do superresolution and the potential improvement of these methods. In Section 3, we propose a new method for superresolution that involves an innovative image acquisition strategy coupled with image processing to achieve the final superresolved image. The resulting technique will yield images whose resolution goes beyond the diffraction limit. In Section 4, we discuss results obtained in our investigation.

2 Background

2.1 The Mathematics of Diffraction-Limited Imaging

To understand the nature of the superresolution problem as well as possible solutions, one needs a strong grasp of the mathematics that describe diffraction-limited imaging. Diffraction-limited imaging refers to the situation in which the aperture is small relative to the radiation wavelength. For simplicity, we consider the imaging equations in one dimension; extension to two dimensions is straightforward. We assume that the source is sufficiently distant from the aperture (in the Fraunhofer or far-field region) so that the wavefront from each radiator can be modeled as a plane wave. Furthermore, we assume that the radiation is essentially monochromatic with wavelength λ . Thus, the plane wave at time t from an angle θ from the aperture plane is $r(p) \exp\{j(\frac{2\pi}{\lambda}(ct + px) + \phi_p)\}$, where $p = \cos\theta$, x is the distance parallel to the aperture plane, and ϕ_p is a phase shift associated with a particular angle of arrival. Consider an aperture centered at $x = 0$. If we assume that the elements of the aperture are insensitive to direction or that the scene exists only for a small angle from perpendicular to the aperture, then the waveform impinging upon the aperture element at a displacement x is a superposition of the waveforms from all angles¹. Thus, we can express the waveform at location x as

$$F(x, t) = S(x) \int_p r(p) \exp\{j(\frac{2\pi}{\lambda}(ct + px) + \phi_p)\} dp, \quad (1)$$

where $S(x)$ is a weighting function that represents the gain of the aperture element at x . Note that this function is simply a weighted Fourier transform of $r(p) \exp\{j(\frac{2\pi}{\lambda}ct + \phi_p)\}$ with respect to p . For future reference, we denote the spatial response transfer function as $H(x)$.

The expression (1) defines the information about the source that is available at the aperture. Since the aperture is spatially limited, the spatial Fourier transform of the scene is only known over specified limits. This frequency-domain truncation is equivalent to a lowpass spatial filter of the image data, which limits the resolution of the image. The shape of the filter impulse response (the point-spread function or PSF) can be controlled by the choice of $S(x)$ ². However, this only changes the weights on the Fourier components; those components that are zero remain zero regardless of the choice of $S(x)$.

As the radiation passes through the aperture, the information is transformed by the intervening lenses and stops. However, since higher spatial frequencies do not generate a response at the aperture, no system of lenses, optical materials, and stops can cause these frequencies to be measured. This is an important point that underscores the difficulty of the superresolution problem. In contrast to many other image deblurring

¹Alternatively, we can replace $r(p)$ with $r(p) \sin\theta$ and then assume that the waveform measured at each aperture element is a superposition of the waveforms from all angles.

²The PSF is also referred to as the beam pattern when it is considered in terms of directional sensitivity at a particular point in the image plane

problems for which higher spatial frequencies are attenuated in the acquired data, in this problem the higher spatial frequencies are eliminated altogether.

The intensity $r^2(p)$ rather than the complex value $r(p) \exp\{j\phi_p\}$ contains the information of interest in passive imaging, since ϕ_p is a random quantity unrelated to the emissivity or temperature of an object. If we denote the PSF of the imaging system as $h(p)$, we can express the intensity of the image at the focal plane as

$$\begin{aligned} |y(p, t)|^2 &= \int_u h(u) r(p-u) \exp\{j\frac{2\pi}{\lambda}ct + j\phi_{p-u}\} du \int_w h^*(w) r(p-w) \exp\{-j\frac{2\pi}{\lambda}ct - j\phi_{p-w}\} dw \\ &= \int_u \int_w h(u) h^*(w) r(p-u) r(p-w) \exp\{j(\phi_{p-u} - \phi_{p-w})\} du dw \end{aligned} \quad (2)$$

The processes that give rise to the radiation are essentially independent from one pixel to the next (noncoherent source), so that the phase ϕ_p is independent and uniformly distributed in $[0, 2\pi]$. Thus,

$$E[\exp\{j(\phi_u - \phi_w)\}] = \delta(u - w) \quad (3)$$

For wideband radiation with bandwidth $\Delta\omega$, the phase terms can be considered independent in time approximately every $\frac{2\pi}{\Delta\omega}$ seconds. Thus, one can estimate the expected value of $|y(p, t)|^2$ by integrating in time:

$$\begin{aligned} \langle |y(p, t)|^2 \rangle &= \lim_{T \rightarrow \infty} \frac{1}{T} \int_0^T |y(p, t)|^2 dt \\ &= E[|y(p, t)|^2] \\ &= \int_u \int_w h(u) h^*(w) r(p-u) r(p-w) \delta(u-w) du dw \\ &= \int_u |h(u)|^2 r^2(p-u) du dw \\ &= |h(p)|^2 * r^2(p) \end{aligned} \quad (4)$$

The result is a convolution of the object intensity distribution with the square of the aperture PSF. The inverse Fourier transform of $|h(p)|^2$ is a convolution of $H(x)$ with itself (assuming a symmetric distribution), which we denote $H_I(x)$. If $H(x)$ has a uniform distribution as shown in Figure 1, then $H_I(x)$ will have a triangular shape with twice the width of $H(x)$ (Figure 2).

Note that the spatial frequencies are attenuated increasingly as x increases. In 1-D the half-power point is at $0.586 \frac{2\pi}{\lambda} x_c$ as compared to $\frac{2\pi}{\lambda} x_c$ for the basic uniform aperture response. In this sense the bandwidth of the intensity imaging scheme is lower than that of the basic aperture spatial frequency response $H(x)$. However, there is an important advantage to the intensity response. Although some higher frequencies are attenuated more than for the uniform response, the intensity imaging system has a nonzero response to some

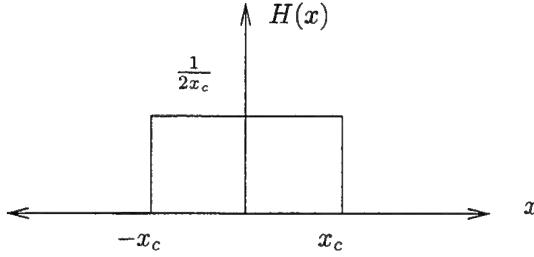


Figure 1: Angular frequency response of uniformly weighted aperture.

spatial frequencies for which the uniform case has zero response. This means that some frequencies that are filtered out by the uniform response are still present for filtering in the intensity case. The presence of these spatial frequencies provides a significant opportunity for image restoration techniques to recover most of the nonzero bandwidth $2\frac{2\pi}{\lambda}x_c$ of the scene.

2.2 Image Processing and Superresolution

Resolution improvement beyond this nonzero bandwidth $2\frac{2\pi}{\lambda}x_c$ is an extremely challenging task. In fact, it is impossible to go beyond this level of resolution with image processing apart from some a priori knowledge of the scene. The sampling theorem implies that the information available at the aperture can be uniquely represented by a set of samples in the spatial image domain with a spacing of $\frac{\lambda}{8\pi x_c}$. Thus, samples spaced more closely than this will be functions of adjacent samples and will not carry any more information about the signal [2].

The sampling theorem assumes that frequencies higher than $2\frac{2\pi}{\lambda}x_c$ are zero. However, if a priori information about the signal is available, one may be able to predict higher spatial frequencies from certain patterns formed by the lower frequencies. If these higher frequencies can be uniquely determined, this in turn specifies a more dense set of spatial-domain samples that are required to represent the larger bandwidth. The prediction of these higher spatial frequencies is the domain of a priori information.

2.2.1 Region of support

A number of assumptions have been used to achieve a measure of superresolution in various settings. One assumption that is sometimes used is that the image has a finite region of support. This assumption has been developed and implemented from a number of different perspectives [3, 4]. Region of support information is quite powerful in the context of astronomical imaging, in which a large part of the scene can be considered to be zero. However, the assumption that the image has a finite region of support is in general inappropriate

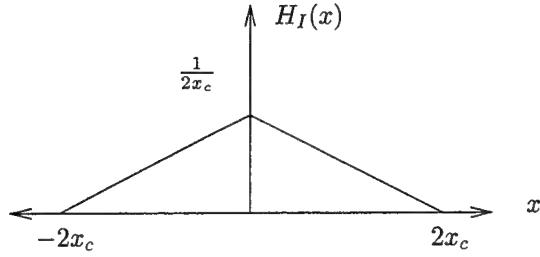


Figure 2: Angular frequency response for amplitude imaging with uniformly weighted aperture.

for the problem considered here.

2.2.2 Nonnegativity

One assumption often used with more success in a general imaging context is that the image is nonnegative. This assumption holds for image intensity, which is normally the quantity of interest in PMMW imaging. The value of this assumption depends primarily on two factors: the amount of noise and the ratio of high-frequency intensity fluctuations to DC intensity. If the noise level is high, infeasible image solutions — including images with negative pixels — are more likely. In that case the nonnegativity constraint will be exercised and will rule out these infeasible solutions. Unfortunately, the nonnegativity constraint contributes nothing to superresolution when noise is the cause of the constraint being exercised. On the other hand, the absence of some high-frequency information may also cause the image solution to go negative. In this case, a nonnegativity constraint will force the missing high-frequency information to appear, which will accomplish a degree of superresolution in the process. Unfortunately, a complex scene's average intensity may be so high that missing high-frequency information does not cause the image to go negative. In this case, a nonnegativity constraint is not exercised at all and thus does not accomplish any superresolution.

2.2.3 Edges

Edges are widely recognized as the basic “building blocks” of an image. Intensity edges generally correspond to object boundaries in a scene, and these are extremely important in both human and computer vision [5]. In fact, some recent work has shown that certain classes of images can be completely reconstructed from knowledge of the edge locations and heights [6, 7]. The informational dominance of edges in real-life images can be used as a priori information in a superresolution scheme. Not surprisingly, a number of techniques have been developed to interpolate images by exploiting edge structure that can be detected in the low-resolution image [8–12]. In essence, each of these methods attempts to preserve the intensity discontinuities of oriented structures (edges). Using any of a number of different mechanisms, the image is interpolated along

directional contours where they exist and is interpolated in a non-oriented fashion where no directionality is evident. These methods significantly improve the quality of interpolated images as compared to standard non-oriented interpolation schemes, such as splines or sinc-function interpolation. Blurring of edges is minimal since interpolation is done only along edges rather than across them.

Unfortunately, these methods have drawbacks as well. Although they reduce (and in some cases eliminate) blurring of edges, they cannot determine the appropriate level of sharpness in a high-resolution edge from a low-resolution representation of the edge. Therefore, the edge may be either too smooth or too sharp in the interpolated image. More seriously, these techniques cannot distinguish edges that are spaced less than a pixel apart. Thus, no new details are created in the interpolation process. While the presence of edges could be an important *a priori* assumption, none of the current techniques appear to be capable of exploiting this assumption to do any significant superresolution.

2.2.4 Few basis images

Another potentially powerful assumption is that the image can be modeled by a limited number of basis images. If these basis images contain frequencies both inside and outside the measured range, the measured frequency components can be used to determine the contribution of each basis image which together implicitly predict the unmeasured frequency components. The challenge is in deciding how to define these basis images so that they have the ability to accurately represent the high-resolution scene. This is useful only in very specialized settings.

2.2.5 Other assumptions

Other structural assumptions, such as the existence of point targets, can also be of value when the assumptions are valid for the data under consideration. However, algorithms that rely on detecting these features in the low-resolution image risk being unable to distinguish overlapping features as well as detecting structure due to noise where no actual structure exists. This risk exists in the constrained iterative deconvolution (CID) algorithm of Richards *et al.* [13] as well as the differentiation-integration deconvolution (DID) algorithm of Ding [14]. Furthermore, these algorithms suffer from a philosophical problem for some applications: if the features of interest can be detected in the low-resolution image, then superresolution is unnecessary. Finally, the assumption of point targets, an integral part of the CID algorithm, is inappropriate for imaging of extended targets.

Unfortunately, for general extended targets, none of the strategies discussed here contribute a significant amount of information about the missing higher frequencies. Our experiments indicate that one can expect only a slight degree of superresolution using these image processing strategies alone [15].

2.3 Image Acquisition

While image processing has the potential to improve the resolution of an image, the degree of potential improvement is quite limited. However, it may be possible to improve the resolution further by adopting new image acquisition strategies. We consider some options here.

2.3.1 Aperture weighting

One can shape the aperture response through $S(x)$ to change the shape of the image PSF. However, as stated earlier, this only changes the relative weights of the measured frequencies; it does not create any additional frequencies. One potential advantage accrues to aperture weighting if the noise is proportional to the signal level rather than at a fixed level. In this case, one can shape the aperture response to improve the superresolution potential by attempting to measure high frequencies at a higher signal-to-noise ratio relative to low frequencies [15]. This makes the data more amenable to restoration; however, it does not allow one to extrapolate new spatial frequencies.

2.3.2 Synthetic aperture imaging

The concept of synthesizing a larger aperture by using a set of smaller apertures is used in a wide variety of settings. Two settings that are closely related to PMMW imaging are radio astronomy [16] and synthetic aperture radar (SAR) [17]. In radio astronomy, a distributed set of fixed apertures is used to improve the resolution of astronomical images. In SAR, the aperture moves along a path perpendicular to the direction to the object and sequentially traces out a distributed aperture. In both cases, coherent summation of the detected signals at each aperture gives the equivalent response of a single larger aperture.

Each case has its own similarities to PMMW imaging. Radio astronomy is similar in that the radiation arises from the scene rather than being transmitted by the observer. SAR is similar in that for most situations where the aperture size is constrained, one must synthesize an aperture from a moving platform. Is it possible to do coherent summation for a noncoherent source? Clearly, that is exactly what is done in radio astronomy. As long as the phase shift from all angles of arrival is small across the entire aperture relative to the coherence length (or time), one can synthesize a larger array. However, it is not clear that one can accomplish this with a single array that must move from one place to another, in spite of such assertions in the literature [18]. The problem is that the coherence time of the radiation is a fraction of $\frac{2\pi}{\Delta\omega}$. If the aperture is moving sideways relative to the scene at 1000 m/s, it will take 0.001 seconds to trace out a 1-meter aperture (in 1-D). If the bandwidth $\frac{\Delta\omega}{2\pi}$ is 10 GHz, the 1-meter aperture trace time is seven orders of magnitude longer than the coherence time of the radiation! Although it may be possible to reduce the bandwidth, a reduction of seven orders of magnitude is probably infeasible. Thus, it appears that SAR is not a viable image acquisition strategy for obtaining PMMW imagery at higher resolutions.

We conclude from our examination of these strategies that current acquisition and processing techniques offer very little in the way of superresolution of PMMW images.

3 Description of Research

We describe a technique that involves a synthesis of a novel acquisition method along with an image restoration scheme to image beyond the diffraction limit.

3.1 Image Acquisition

Earlier we observed that by averaging intensity over time, the PSF becomes $|h(p)|^2$ instead of $h(p)$. The squared PSF has a Fourier transform that is twice as wide as the original PSF Fourier transform, which allows for the possibility of achieving twice the resolution obtainable with the original PSF. We have discovered that the concept of raising the amplitude to a higher power can be extended to powers greater than two. In fact, if we average the squared intensity over time, we get the following:

$$\begin{aligned}
 \langle |y(p, t)|^4 \rangle &= \lim_{T \rightarrow \infty} \frac{1}{T} \int_0^T |y(p, t)|^4 dt \\
 &= E[|y(p, t)|^4] \\
 &= \int_a \int_b \int_u \int_w h(a)h^*(b)h(u)h^*(w)r(p-a)r(p-b)r(p-u)r(p-w) \\
 &\quad \times [\delta(a-b)\delta(u-w) + \delta(a-w)\delta(b-u) - \delta(a-b)\delta(u-w)\delta(a-u)] da db du dw \\
 &= 2 \left[\int_u |h(u)|^2 r^2(p-u) du \right]^2 - \int_u |h(u)|^4 r^4(p-u) du \\
 &= 2[|h(p)|^2 * r^2(p)]^2 - |h(p)|^4 * r^4(p)
 \end{aligned} \tag{5}$$

The fourth-order image is then a function not only of the low-resolution intensity image but also a higher-resolution intensity-squared image. By subtracting a measured version of $2[|h(p)|^2 * r^2(p)]^2$ ($= 2 \langle |y(p, t)|^2 \rangle^2$), one can isolate the term $|h(p)|^4 * r^4(p)$. The issues associated with the use of the intensity image also apply to this image. First, the convolution with $|h(p)|^4$ may attenuate higher frequencies even faster than $|h(p)|^2$. However, the Fourier transform $H_{I^2}(x)$ is also twice as wide, meaning that the measured bandwidth is twice that of the intensity image. (See Figure 3.) Thus, the measured fourth-order data has the potential for superresolution beyond the diffraction limit since these higher frequencies still exist in the data. Second, the expected value is only approximately achieved in a finite amount of time. Consequently, this model will have a noise variance that is inversely related to the time available for acquisition. Third, one can design the aperture weighting $S(x)$ to maximize SNR in the acquired image with the goal of superresolution in view.

To see that $\langle |y(p)|^4 \rangle$ does contain higher frequencies of the intensity $r^2(p)$ and not simply of $r^4(p)$,

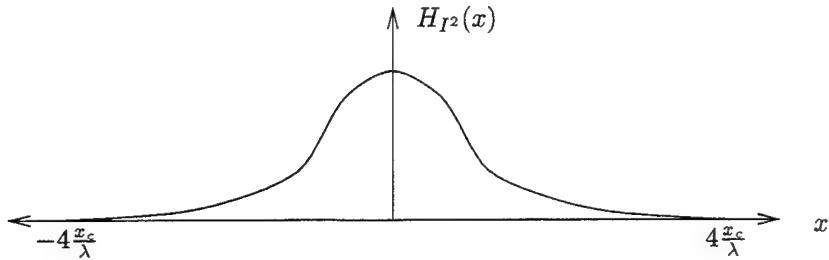


Figure 3: Angular frequency response for intensity-squared imaging with uniformly weighted aperture.

consider a Taylor series expansion of $f(x) = x^2$ around $x_0 = \overline{r^2(p)}$ (the mean value of the scene intensity), where $x = r^2(p)$:

$$r^4(p) \approx \left[\overline{r^2(p)} \right]^2 + 2\overline{r^2(p)}(r^2(p) - \overline{r^2(p)}) \quad (6)$$

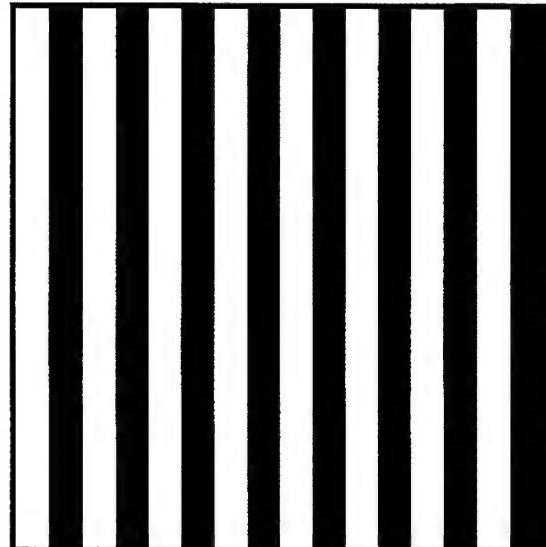
Convolving with $|h(p)|^4$ yields the form

$$|h(p)|^4 * r^4(p) \approx K_1 + K_2 |h(p)|^4 * r^2(p) \quad (7)$$

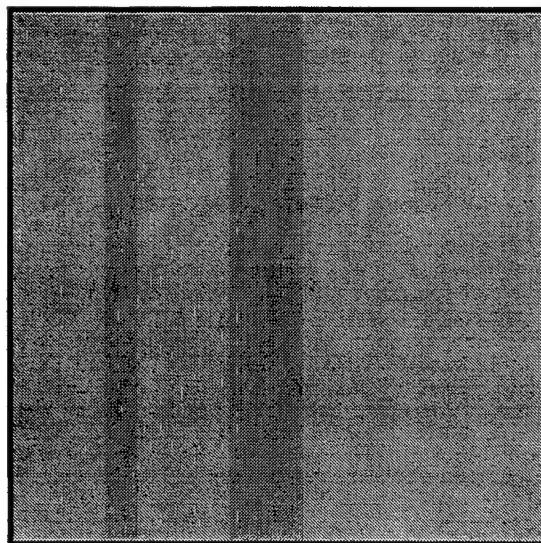
where K_1 and K_2 are constants. The term $|h(p)|^4 * r^2(p)$ measures twice the bandwidth of $r^2(p)$ as does the term $|h(p)|^2 * r^2(p)$.

To verify the validity of the fourth-order acquisition method, we generated a 64-point square wave (a bar pattern in 2-D) and simulated an intensity image as well as an intensity-squared image. The signal was replicated 64 times to form a 64×64 image for display purposes (Figure 4(a)). The square wave was used as the amplitude with the phase at each sample chosen randomly. The aperture size was chosen so that the spatial frequency of the square wave was higher than the highest measured frequency in the intensity image. Thus, one would expect a response only at DC in the intensity image. An intensity image and an intensity-squared image were formed by a Monte Carlo simulation taking 100,000 independent phase realizations and averaging the resulting signals. The resulting intensity signal is shown in 4(b) and the intensity-squared signal in 4(c). Note that the intensity signal has only a DC component as expected. However, the intensity-squared signal shows a frequency component at the same frequency as the square wave. Of course, this signal is blurred since even the intensity-squared signal does not pick up all frequencies of the square wave. This simulation confirms that higher frequencies can be imaged by the higher-order imaging scheme that are completely missed by a conventional intensity imaging scheme.

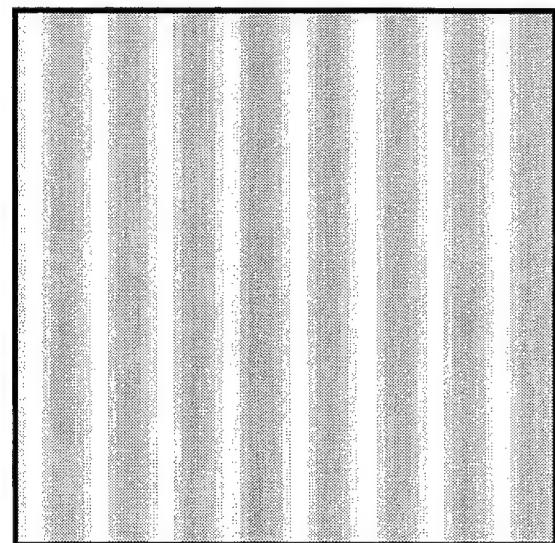
The concept behind the use of fourth-order images can be generalized to higher orders with a consequent increase in the measured bandwidth. Theoretically, one can continue this process to achieve any desired bandwidth (and resolution). Unfortunately, as the order is increased, the assumption that the expected



(a)



(b)



(c)

Figure 4: Higher-order imaging experiment: (a) original signal, (b) signal acquired by averaging intensity after passing through aperture, (c) signal acquired by averaging squared intensity after passing through aperture

value is achieved for a given amount of time becomes an increasingly poor assumption. Thus, noise becomes much more serious as the order is increased.

3.2 Analytical results

We have shown that the variance of the intensity image is given by

$$\frac{\tau}{T} \langle |y(p,t)|^4 \rangle, \quad (8)$$

where τ is the coherence time of the radiation (approximately $\frac{1}{BW}$ of the radiation detected at the aperture) and T is the integration time.

Likewise, the variance of the intensity-squared image is given by

$$\frac{\tau}{T} \langle |y(p,t)|^8 \rangle, \quad (9)$$

This implies that brighter areas of the image will have a larger variance than those that are less bright. In other words, the variance is spatially dependent.

The fourth-order signal is strictly bandlimited by the finite aperture both for finite and infinite integration time. Therefore, the noise due to finite integration time (the discrepancy between the measured signal and the mean value) must also be strictly bandlimited to the same degree. Work is ongoing to further characterize the nature of this bandlimited noise.

Our simulations indicate that an integration time on the order of one second will be necessary to reduce the noise to a visually insignificant level. This places a limit on the potential applicability of this technique apart from a mult-frame approach in which prior frames are used as side information in the postprocessing stage.

3.3 Image processing

To exploit higher-order imaging, one must determine how to combine the information in the second-order image (the intensity image) with the information in the fourth-order image. One approach to this problem is to follow a Taylor series strategy similar to (6). Let $\widehat{r^2(p)}$ be the best current estimate of the intensity image. Then expand $f(x) = x^2$ around $x_0 = \widehat{r^2(p)}$, where $x = r^2(p)$:

$$r^4(p) \approx \left[\widehat{r^2(p)} \right]^2 + 2\widehat{r^2(p)}(r^2(p) - \widehat{r^2(p)}) \quad (10)$$

Convolving with $|h(p)|^4$ yields the form

$$|h(p)|^4 * r^4(p) \approx K_1(p) + |h(p)|^4 * K_2(p)r^2(p) \quad (11)$$

where $K_1(p)$ and $K_2(p)$ are functions of p . This expansion suggests an algorithmic approach to estimating the intensity image from the fourth-order data — seek to minimize the squared difference between the measured data representing $|h(p)|^4 * r^4(p)$ and the approximate representation on the right side of (11) as a function of $r^2(p)$. In addition to the fourth-order image, the measured second-order image contains information about $r^2(p)$. Thus, we can seek to minimize a weighted combination of squared difference measures in both measured sets of data to improve the fidelity of the result.

An iterative algorithm can be formulated using the result (11). However, iterative algorithms are generally quite computationally intensive and may not be as appropriate for real-time imaging without significant computational power. Instead of an iterative approach, we developed a straightforward noniterative algorithm to superresolve the image.

In contrast to the previous discussion, we describe the algorithm using two-dimensional discrete notation, since this is actually how it was implemented. The algorithm works as follows. Let $i(m, n)$ represent the finite-time integrated intensity image after discretization and $s(m, n)$ represent the finite-time integrated squared-intensity image after discretization. Furthermore, let $h^4(m, n)$ represent the discretized version of the PSF of the fourth-order process.

Form the pseudo-data $d(m, n)$ as

$$d(m, n) = 2[i(m, n)]^2 - s(m, n) \quad (12)$$

This yields a data set whose image formation equation is a simple convolution with additive noise:

$$d(m, n) = h^4(m, n) * r^4(m, n) + u(m, n), \quad (13)$$

where $u(m, n)$ is noise representing observation noise and finite integration-time noise.

A regularized inverse yields an approximation of the superresolved intensity-squared image $r^4(m, n)$:

$$\widehat{r^4(m, n)} = \mathcal{F}^{-1} \left\{ \frac{H_4^*(k, l)}{|H_4^*(k, l)|^2 + K} D(k, l) \right\}, \quad (14)$$

where $D(k, l)$ is the FFT of $d(m, n)$, $H_4(k, l)$ is the FFT of $h^4(m, n)$, \mathcal{F}^{-1} is the inverse FFT, and K is some predefined constant. The constant K is chosen to optimize the tradeoff between restoration and amplification of noise.

Finally, we take a square root of the result in (14) to yield the superresolved image. Since some elements may be negative, we first set them to zero before taking the square root:

$$\widehat{r^2(m,n)} = \sqrt{\max\{\widehat{r^4(m,n)}, 0\}} \quad (15)$$

3.4 Experiments

We chose the building image in Figure 5(a) to represent the amplitude image. A circular aperture equivalent to a discrete frequency diameter of $\frac{9}{256}$ was chosen. Data was simulated for an infinite imaging time for both second-order (intensity) data and fourth-order (intensity-squared) data. Figures 5(b) and 5(c) show these images. The image was restored using the method described in the previous section with K set approximately four orders of magnitude lower than the peak value of $|H_4(k,l)|^2$. The restored image is shown in Figure 5(d). For comparison purposes the restored intensity image is shown in 5(e). Note that the image using the fourth-order data is clearly superresolved in relation to both the unrestored data 5(b) and restored intensity data 5(e).

In addition, the effective aperture for the superresolution system was calculated as follows: First, a unit impulse was added to the original building image, and the infinite-time data was simulated. This data was restored using the technique described previously. Then the result of the superresolved building was subtracted from the restoration with the impulse added. This yielded an estimate of the point-spread function of the system. An FFT was taken of this, and the resulting frequency response showed an effective discrete-frequency radius of $\frac{27}{256}$. This indicates a superresolution ratio of 3 when the original data is used as the standard.

4 Conclusions

Our preliminary investigation showed that the proposed method for superresolution does yield superresolved images. Our experiments indicate that we can obtain an improvement by a factor of 1.5 beyond that which can be obtained by postprocessing an intensity image alone. The postprocessing method described here has the advantage of being quite simple and computationally light, but it is far from optimal in using the information available to the problem. With more sophisticated image processing, better results are probable.

In our experiments, we did not take into account the effect of finite integration time. This may ultimately be the limiting factor for many applications. The fourth-order image requires a great deal more integration time to closely approximate the expected value (to reduce the approximation noise) than the second-order image. Work is ongoing to determine exactly what the requirements are in this regard and to develop more effective methods for acquiring, combining, and processing the data to yield better images.

References

- [1] R. Appleby, D. G. Gleed, and R. N. Anderton, "High-performance passive millimeter-wave imaging," *Optical Engineering*, vol. 32, pp. 1370–1373, June 1993.
- [2] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. New Jersey: Prentice-Hall, 1989.
- [3] J. W. Goodman, *Introduction to Fourier Optics*. San Francisco: McGraw-Hill, 1968.
- [4] R. W. Gerchberg, "Super-resolution through error energy reduction," *Optica Acta*, vol. 14, no. 9, pp. 709–720, 1979.
- [5] J. Aloimonos and D. Shulman, "Learning early-vision computations," *Journal of the Optical Society of America A: Optics and Image Science*, vol. 6, pp. 908–919, June 1989.
- [6] R. Alter-Gartenberg, F. O. Huck, and R. Narayanswamy, "Compact image representation by edge primitives," *Journal of the Optical Society of America A*, vol. 7, pp. 898–911, May 1990.
- [7] R. Alter-Gartenberg, F. O. Huck, and R. Narayanswamy, "Compact image representation by edge primitives," *CVGIP: Graphical Models and Image Processing*, vol. 56, pp. 1–7, January 1994.
- [8] K. Jensen and D. Anastassiou, "Spatial resolution enhancement of images using nonlinear interpolation," in *Proceedings of the 1990 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 2045–2048.
- [9] V. R. Algazi, G. E. Ford, and R. Potharlanka, "Directional interpolation of images based on visual properties and rank order filtering," in *Proceedings of the 1991 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 3005–3008.
- [10] R. H. Bamberger, "A method for image interpolation based on a novel multirate filter bank structure and properties of the human visual system," in *SPIE Vol. 1657 — Image Processing Algorithms and Techniques III*, pp. 351–362, 1992.
- [11] G. E. Ford, R. R. Estes, and H. Chen, "Space scale analysis for image sampling and interpolation," in *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. III-165–III-168.
- [12] R. R. Schultz and R. L. Stevenson, "A Bayesian approach to image expansion for improved definition," *IEEE Transactions on Image Processing*, vol. 3, pp. 233–242, May 1994.
- [13] M. A. Richards, C. E. Morris, and M. H. Hayes, "Iterative enhancement of noncoherent radar data," in *Proceedings of the 1986 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 1929–1932.
- [14] Z. Ding, "Resolution enhancement of passive millimeter-wave imaging." Final Report: Summer Faculty Research Program, AFOSR, 1993.
- [15] S. J. Reeves, "Superresolution of passive millimeter-wave imaging." Final Report: Summer Faculty Research Program, AFOSR, 1994.
- [16] A. R. Thompson, J. M. Moran, and G. W. Swenson, Jr., *Interferometry and Synthesis in Radio Astronomy*. New York: John Wiley & Sons, 1986.
- [17] D. L. Mensa, *High Resolution Radar Imaging*. Dedham, MA: Artech House, 1981.
- [18] K. Komiyama, "High resolution imaging by supersynthesis radiometers (SSR) for the passive microwave remote sensing of the Earth," *Electronics Letters*, vol. 27, p. 389ff., February 1991.

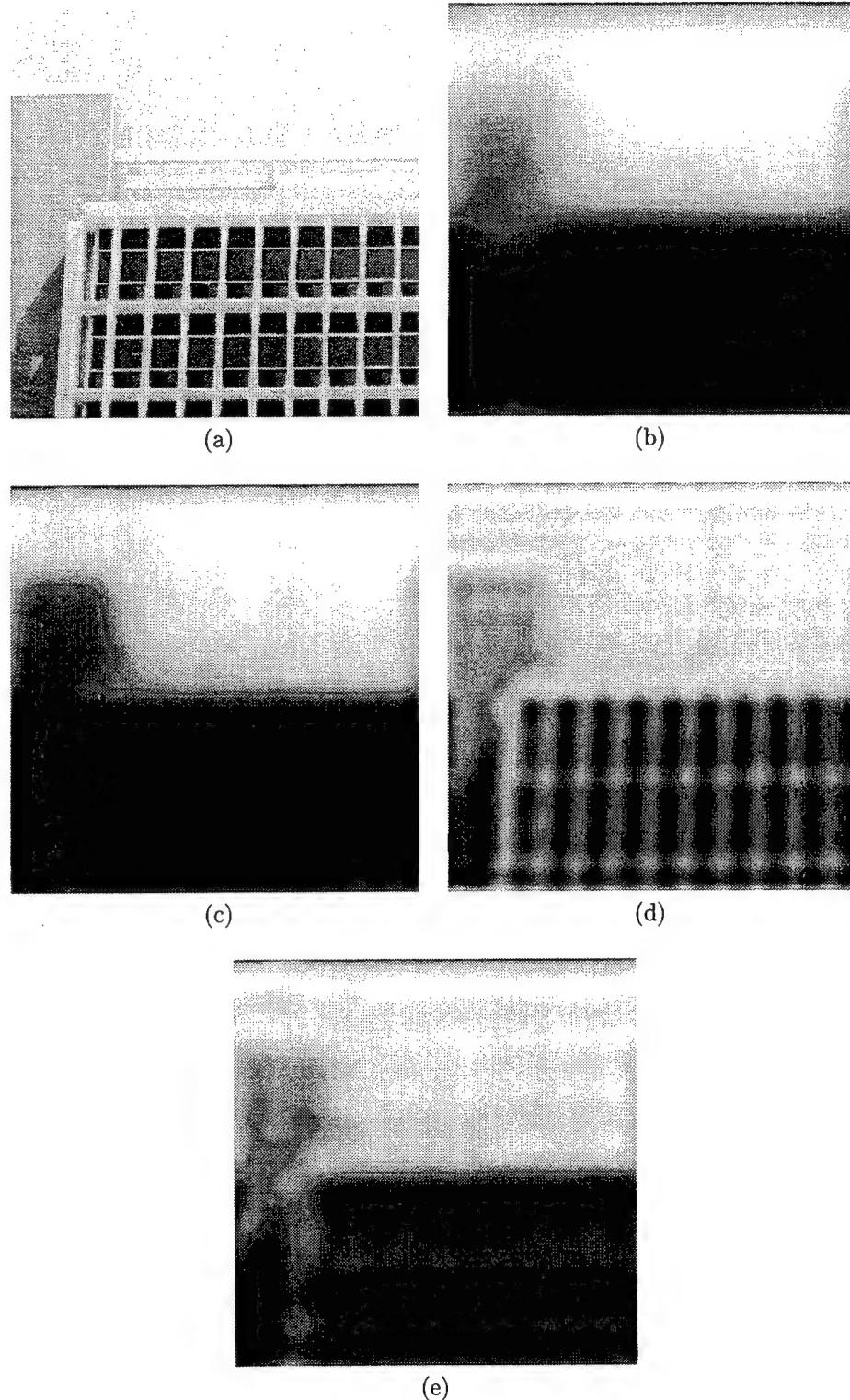


Figure 5: (a) Original amplitude image, (b) integrated intensity image, (c) integrated fourth-order image, (d) restored image using proposed superresolution method, (e) restored version of (b).

**TAYLOR - A PREPROCESSOR, POSTPROCESSOR, AND OPTIMIZER FOR NUMERICALLY
MODELING TAYLOR IMPACT SPECIMENS WITH EPIC**

William K. Rule

Associate Professor

Department of Engineering Science and Mechanics

The University of Alabama

Box 870278

Tuscaloosa, Alabama 35487-0278

Final Report for:

Summer Faculty Research Program

Wright Laboratory

Sponsored by:

Air Force Office of Scientific Research

Bolling Air Force Base, DC

and

Wright Laboratory

December 1995

**TAYLOR - A PREPROCESSOR, POSTPROCESSOR, AND OPTIMIZER FOR NUMERICALLY
MODELING TAYLOR IMPACT SPECIMENS WITH EPIC**

William K. Rule
Associate Professor
Department of Engineering Science and Mechanics
The University of Alabama

Abstract

Computer simulations of high velocity impact events provide a cost effective means for analyzing weapon and armor systems. However, high velocity impact simulation codes require complicated and comprehensive material property input data and are difficult to use by those who are not specialists in computational mechanics such as experimentalists and designers. This project attempts to address some aspects of these two issues.

Over the last 50 years Taylor testing has been used to measure the dynamic yield strength of materials. Recently, this test has been used to determine or verify coefficients for formulas such as the Johnson-Cook strength model. Taylor testing involves launching a cylinder of the material of interest at an essentially rigid anvil. Using black powder guns, launch speeds are typically of the order 200 m/s and strain rates approaching $10^5/s$ can be produced in the impacted end of the specimen. The deformed geometry of the tested specimen is used to infer the dynamic material properties.

This project involved developing a computer program called TAYLOR to facilitate the numerical modeling of Taylor specimens using the finite element code EPIC. Seamlessly, TAYLOR can generate an EPIC input deck (preprocess), can run the EPIC code, can display the calculated results (postprocess), and can optimize material strength model coefficients to minimize discrepancies with experimental data. TAYLOR provides experimentalists with a high fidelity numerical model that they can use to assist in the study of the physical processes that occur within a deforming Taylor specimen. TAYLOR can also be used to verify the accuracy of strength model coefficients obtained from other sources of experimental data. This can be accomplished by comparing measured and calculated (from EPIC run by the TAYLOR program) Taylor specimen deformed profiles. Finally, the optimizer of TAYLOR can be used to fit strength model coefficients using Taylor test data.

TAYLOR is written in Visual BASIC (Ver. 3), and is designed to run under Microsoft Windows 3.1 on a personal computer. The version of EPIC that TAYLOR runs is a slightly modified form of Research EPIC 95 that was compiled as a DOS executable using Microsoft FORTRAN PowerStation (Ver. 1). This version of TAYLOR, except for the optimizer, will also run reasonably effectively under Microsoft Windows 95. The TAYLOR user requires no detailed knowledge of EPIC, Visual BASIC, or FORTRAN.

TAYLOR - A PREPROCESSOR, POSTPROCESSOR, AND OPTIMIZER FOR NUMERICALLY MODELING TAYLOR IMPACT SPECIMENS WITH EPIC

William K. Rule

1. Introduction

Taylor impact testing is a very commonly used technique to obtain information about the behavior of materials subjected to high strain rates. Taylor testing involves launching a cylindrical specimen of the material of interest at an essentially rigid target. This experimental technique is discussed in Section 2. The experimental data typically obtained from Taylor testing consists of high speed photographs of the deforming specimen as a function of time from impact with the target, and the final deformed shape of the specimen. While this data is very useful, it is desirable to obtain additional information about the mechanical processes taking place inside the specimen during the impact event. Such full field data can provide important insights on the impact physics.

One common way to get full field data about a deforming Taylor specimen is to use a calibrated numerical model. The numerical model considered for this study was a finite element code called EPIC. EPIC was first developed two decades ago (Johnson, 1977) and is still being enhanced (Johnson, 1994). Many other numerical models are available for performing this task, some of which use completely different numerical strategies such as the finite difference approach (Karpp, 1993).

Experimental testing and numerical modeling at high strain rates are very different and complex fields of knowledge. It is difficult for experimentalists to develop, with confidence, numerical models for their specimens. The purpose of this project was to develop special purpose software (called TAYLOR¹) to easily allow experimentalists to numerically model their Taylor specimens using EPIC with virtually no knowledge of the finer technical details of this code. As is described in more detail in subsequent sections, TAYLOR seamlessly builds the input file for EPIC (acts as a preprocessor), launches EPIC, and then graphically displays the results of the EPIC analysis (acts as a postprocessor). In addition, TAYLOR has an easy-to-use optimizer function that automatically adjusts material strength model coefficients to minimize the discrepancy between calculated and measured specimen behavior. Also, TAYLOR's data files provide a convenient means for storing strength model coefficients in a single place.

TAYLOR is written in Visual BASIC (Ver. 3) and runs under Microsoft Windows (Ver. 3.1) on a personal computer. TAYLOR launches (a slightly modified version of) the 1995 research edition of EPIC. EPIC was compiled using Microsoft FORTRAN PowerStation (Ver. 1) for running on a PC in conjunction with TAYLOR. Typical EPIC runs launched by TAYLOR conveniently take only 5-15 minutes to run (depending on the

¹ In this report TAYLOR refers to the computer code, while Taylor refers to the experimental test.

finite element mesh density) on a 100 MHz Pentium class computer. Section 5 describes steps to be taken to get most of the functionality of TAYLOR while running under Windows 95.

2. Taylor Impact Testing

The Taylor impact test was developed fifty years ago by Sir Geoffrey Taylor (1948) to predict the dynamic yield stress of materials subjected to high strain rates. Rakhmatulin (1945) also developed a similar test during the same time period. The test is conducted by launching a cylinder of the material of interest at a flat, rigid target at speeds sufficiently high to generate the strain rates of interest, which can approach $10^5/\text{s}$. Taylor used the final deformed shape of the specimen to determine the dynamic yield strength (stress) of the material. Taylor's analytic theory has since been modified by many researchers (Lee and Tupper, 1954; Hawkyard, 1969; Jones et al., 1987).

A typical Taylor test setup is shown in Fig. 2.1² (House, 1989; Allen, 1995). The Taylor test provides a very safe and cost effective method for generating high strain rates. The equipment is capable of generating reproducible results. The smooth-bored gun barrels used are typically 25 in. long and calibers ranging from .17 in. to 0.50 in. have been used successfully. The targets are usually constructed of specially hardened steel and the impact face is polished to a mirror-like finish to minimize frictional effects on the Taylor specimen. The targets are designed to weigh at least 80 lb. to ensure that they move very little on impact in order to properly simulate a rigid surface. The target is usually rotated before each shot so that impact will occur on a smooth, unblemished surface.

The Taylor specimens are launched by igniting a measured amount of commercial grade gunpowder that was placed in a conventional primed shell casing. The shell casing primer is detonated by striking it with a firing pin driven by a remotely operated solenoid. The specimen velocity can be easily controlled by adjusting the amount of gunpowder placed in the shell casing. Powder curves, where specimen velocity versus powder mass is plotted, can be easily created and used to estimate the powder required to generate the desired specimen velocity.

Specimen velocities can be measured by the use of pressure transducers along the gun barrel (House, 1989) or by the use of a laser system (Allen, 1995) as illustrated in Fig. 2.1. The deforming specimen can be captured on film with a high speed camera (House, 1989) and the final plastically-deformed geometry of the specimen can be measured by calipers or by using an optical comparitor (Allen, 1995). The optical comparitor functions by projecting a magnified image of the specimen on a specially scaled viewing screen. The enlarged image allows for accurate measurements to be taken of the deformed diameter of the specimen as a function of distance from the impacted end.

3. The EPIC Finite Element Code

This project involved the development of software to make it relatively easy for experimentalists and others not familiar with computational mechanics to numerically analyze Taylor impact specimens with EPIC. The development of EPIC began two decades ago (Johnson, 1977) and since then the code has become increasingly

² The figures have been collected together at the end of this report.

complex. Recent publications by Karpp (1993) and Johnson (1993) describe computational impact mechanics in general and EPIC in particular, respectively.

It is relatively easy to derive the governing differential equations that describe the behavior of a deforming solid body. The difficulty is finding solutions to these differential equations that are compatible with a general set of boundary conditions. This obstacle can be overcome by considering the body to be subdivided into a number of small, but finite zones, called finite elements. Typical element shapes consist of triangles and tetrahedrons for two-dimensional and three dimensional problems, respectively. The corners of the elements are defined by points in space called nodes. Additional nodes are sometimes placed along the sides of the element to obtain more accurate results at the expense of increased computation time. The elements are set up in a regular fashion so that it is possible to obtain a solution to the governing equations for each element by itself. A solution to the problem as a whole can be obtained by forcing each element to deform in a manner that is compatible with attached neighboring elements. A large number of different types of elements for various purposes have been developed. The TAYLOR program uses three-noded, triangular, axisymmetric elements to exploit the cylindrical symmetry of the Taylor specimen. As described below, the TAYLOR user can control the number of nodes and elements used in the numerical model. Figure 3.1 shows a typical deformed finite element mesh of a Taylor specimen. EPIC is capable of outputting nodal positions and velocities at user specified time intervals.

It is difficult to decide how dense a finite element mesh is required to produce results of sufficient accuracy for the intended application. As more elements are used accuracy generally improves but execution times increase. The usual approach is to start with a coarse mesh and make successive runs with progressively finer meshes until the calculated property of interest (deformations, stresses, strains, failure index, and so forth) converges.

The elements of the Taylor specimen must not be allowed to penetrate through the target interface, which is assumed to be rigid, smooth, and frictionless. EPIC has special provision for this type of boundary condition and the TAYLOR program invokes it.

During an impact event very large pressures are generated along with extensive plastic flow. EPIC (and many other codes) treats these two characteristics separately. To model the pressure, P , the Mie-Gruneisen equation of state is used by EPIC (Johnson, 1993):

$$P = \left(K_1 \mu + K_2 \mu^2 + K_3 \mu^3 \right) \left(1 - \Gamma \frac{\mu}{2} \right) + \Gamma E_s (1 + \mu) \quad (3.1)$$

In equation (3.1), $\mu = \frac{\rho}{\rho_0} - 1$ (where ρ_0 and ρ are the initial and subsequent densities, respectively), represents the degree to which the material is being compressed. E_s is the internal energy, and K_1 , K_2 , K_3 , and Γ (Gruneisen coefficient) are coefficients that depend on the material used. Note that at relatively low pressures (small μ) K_1

functions like a bulk modulus. These material property coefficients are almost completely dependent on the main component atom of a material and thus vary very little from one alloy to the next. EPIC also requires the input of the maximum hydrostatic tensile pressure (stress) allowed (parameter PMIN). The user can specify the time intervals at which the pressure (on an element by element basis) should be output.

However, plastic flow behavior is, of course, very dependent on the particular alloy under consideration. Thus, determining material property coefficients to treat plastic flow behavior for new materials is more difficult than is the case for pressure modeling. The Johnson-Cook (1983) yield strength function is often used to determine the stress level, σ , at which plastic flow will initiate:

$$\sigma = (C_1 + C_2 \varepsilon^N) (1 + C_3 \ln \dot{\varepsilon}^*) (1 - T^{*M}) \quad (3.2)$$

where the variables ε , $\dot{\varepsilon}^*$, and T^* are the effective plastic strain, the effective plastic strain rate, and the homologous temperature, respectively. The material constants C_1 , C_2 , N , C_3 , and M are must be determined from experiments or estimated by comparison with the material constants of similar materials. Section 9 explains how the TAYLOR program can be used to automatically fit yield strength coefficients from Taylor test data. The Johnson-Cook model also allows a maximum allowable yield stress (SMAX) to be specified.

The TAYLOR program also supports the other strength models contained in the EPIC code: modified Johnson-Cook (Holmquist and Johnson, 1991), Zerilli-Armstrong (FCC and BCC, Zerilli and Armstrong, 1987), Bodner-Partom (Bodner and Partom, 1975; Bodner and Merzer, 1978; Cook et al., 1992), and MTS (Follansbee and Kocks, 1988; Follansbee and Gray, 1989).

EPIC can output the effective (or von Mises) stress, effective plastic strain, the effective plastic strain rate, and the temperature, on an element by element basis, at user selected time intervals.

Modeling the failure of materials under high loading rates is very difficult and currently much research is devoted to this task. EPIC contains a reasonably reliable formula (Johnson and Cook, 1985) for predicting the strain level, ε^f , at which material failure will occur:

$$\varepsilon^f = (D_1 + D_2 e^{D_3 \sigma^*}) (1 + D_4 \ln \dot{\varepsilon}^*) (1 + D_5 T^*) \quad (3.3)$$

In equation (3.3), σ^* is the ratio of the mean of the normal stresses to the von Mises equivalent stress, and $\dot{\varepsilon}^*$ and T^* are as was defined previously.

The coefficients D_1 through D_5 in equation (3.3) are obtained from testing to failure the material of interest. EPIC also requires two additional empirical coefficients for material failure modeling: σ_{spall} , a coefficient related to the stress at which failure can occur, and ε_{\min}^f , the minimum allowable fracture strain. EPIC has a scheme (which uses the parameter SOFT, with $0 \leq \text{SOFT} \leq 1$) for allowing the material to soften gradually instead of failing abruptly. Abrupt failure is modeled by setting $\text{SOFT} = 0$. EPIC reports the level of damage on an element by element basis in terms of the damage fraction D , where $0 \leq D \leq 1$. A D value of unity implies that the material of that element has failed.

Shock waves travel through materials subjected to sudden loads. Modeling shock waves numerically is difficult because at the shock front the system variables (such as pressure) change very suddenly and are almost discontinuous. Large spurious oscillations appear in the numerical results near compressive shock fronts unless a special correction technique, called artificial viscosity, is included in the model. EPIC uses both linear and quadratic artificial viscosity terms. The magnitudes of these correction terms are controlled by EPIC input parameters CL (0.2) and CQ (4.0), for the linear and quadratic terms, respectively. Recommended values of these parameters are given in the brackets.

For 2D quad (four sided) and 3D brick elements a numerical mesh instability problem called hourgassing can occur. This problem is corrected by an hourglass artificial viscosity coefficient the magnitude of which is set by the EPIC input parameter CH. TAYLOR does not use these types of elements and so the magnitude of CH is of no concern here.

4. Modifications Made to EPIC

EPIC is designed to output calculated results at time intervals specified by the user. The calculated results are in the form of nodal and elemental data. Various output formats compatible with the commercial MSC/PATRAN postprocessor package can be requested by the EPIC user (PAT numbers 1-5). For this project, due to file size limitations for convenient manipulations on the personal computer, it was decided to create new, compact output file formats especially for use by TAYLOR. To accomplish this some minor modifications were made to EPIC, but in such a way that the standard capabilities of the code were not affected. The special data files are requested by specifying a PAT number of 6 in the EPIC input file. The TAYLOR program does this automatically. However, simply changing the PAT number from 6 to a number in the conventional range (1-5) will allow the TAYLOR generated input file to be used with conventional versions of EPIC on other computer platforms. This would be necessary, for instance, if the TAYLOR program user wishes to employ more than 2000 nodes or 4000 elements, which are the maximum array sizes set in the version of EPIC bundled with TAYLOR.

One special data file contains nodal data and is called NODEDATA.TXT. The first line of this file lists the problem description and then a series of data sets (one for each output time requested). The first line of each data set contains the simulation time and the number of nodes. Following this line the nodal data is listed, one line

for each node. The nodal data consists of x, y, z coordinates and velocities. TAYLOR builds axisymmetric models so the y data is ignored and the x data corresponds to the radial direction. The element data file (ELEMDATA.TXT) is of the same format as the nodal data file except that the number of elements are given at the start of each data set. The element data listed (one line for each element) are: pressure, von Mises stress, effective strain, damage index, temperature, and strain rate.

5. Starting the TAYLOR Program

5.1 Starting TAYLOR Under Microsoft Windows 3.1

The files of TAYLOR are conveniently contained on two floppy disks. TAYLOR can be installed by simply copying the contents of these disks into a subdirectory of the computer's hard drive. The program is started by double clicking on the executable (TAYLOR.EXE) under File Manager or by any other Windows 3.1 program launching technique. The meanings of the various command buttons, input boxes, and options of TAYLOR are described in subsequent sections of this report.

5.2 Starting TAYLOR Under Microsoft Windows 95

This version of TAYLOR was designed to run under Windows 3.1. Windows 3.1 is essentially an operating system shell that sits on top of the main operating system - which is DOS. Thus it is relatively easy to switch between Windows and DOS programs while running under Windows 3.1. This is a requirement for running the TAYLOR program since the compiled version of EPIC is actually a DOS program. Microsoft does not market a FORTRAN compiler to create Windows 3.1 executables. Windows 3.1 allows users to launch a DOS program without shutting down the Windows shell.

Although this Windows-DOS switching capability is often convenient, having Windows sitting on top of DOS greatly constrains the capabilities and the speed of the Windows shell. Accordingly, in the latest version of the software, Windows 95, DOS is essentially bundled as a separate operating system and the Windows part no longer sits on DOS. Running a large DOS program (such as EPIC) requires that the Windows portion of the software be completely shut down and restarted after execution is completed. This means that the optimizer portion of TAYLOR will not work under Windows 95 until EPIC can be recompiled using a FORTRAN compiler that produces Windows 95 executables. The writer has just obtained such a software package and so a fully functional Windows 95 version of TAYLOR should be available shortly.

To launch EPIC under Windows 3.1 (which TAYLOR does automatically) a special Windows-DOS program interface file (.PIF) was created. This file must be adjusted for running under Windows 95. This can be accomplished by selecting the .PIF file, and then under the FILE menu of a Windows 95 file manager click on PROPERTIES, then click the PROGRAM TAB, then click the ADVANCED BUTTON, and then click MS-DOS mode option. This will reset the .PIF file to be compatible with the requirements of Windows 95.

6. The Preprocessor of the TAYLOR Program

The preprocessor of TAYLOR is designed to create an input file for EPIC. This involves specifying: the specimen size, the finite element mesh density, the dynamic material properties, the impact velocity, and output time interval requirements. How TAYLOR facilitates this process will now be described.

The main TAYLOR program window is shown in Fig. 6.1. The Taylor specimen Length and Radius are entered in the text boxes in the left-central portion of the main program window, Fig. 6.1. The mesh density is selected by clicking on the appropriate option button in the central portion of the main program window. Default Coarse, Medium and Fine mesh options have been set up for the convenience of the user. If these mesh configurations are deemed unacceptable then clicking the Custom mesh option button will cause the window of Fig. 6.2 to pop open allowing the user to define the mesh density. The selected mesh is displayed to scale in the bottom-left corner of the main program window. In keeping with the axisymmetric nature of the model, only half the mesh cross section is displayed.

TAYLOR uses two material property data files. One file (LIB_MAT.TXT) contains material property data for those standard materials bundled with Research EPIC 95. The other material property data file (MATERIAL.TXT) contains properties either entered by the TAYLOR user or produced by the TAYLOR optimizer (see Section 9). MATERIAL.TXT also contains a copy of the contents of LIB_MAT.TXT to provide convenient defaults while specifying custom material properties (as discussed below). These ASCII data files can be modified outside of TAYLOR with a text editor such as NOTEPAD. TAYLOR also provides for adding and removing records from MATERIAL.TXT as discussed below.

The contents of both files are listed in the **Select Specimen Material** box in the upper-left corner of the main program window, Fig. 6.1. Non-standard materials are listed with a * before their name. All available materials in the data files can be viewed by exercising the scroll bar along the right side of the **Select Specimen Material** box. A listed material may be selected by simply clicking on the name of the desired material which will cause it to become highlighted.

New sets of material properties can be specified by first clicking on the **Manage Custom Materials** button at the bottom-left corner of the main program window, Fig. 6.1. This will cause the window of Fig. 6.3 to be displayed. Selecting **Add New Custom Material Properties to Data File MATERIAL.TXT** and clicking the **Continue** button will cause the general material data window to pop open, Fig. 6.4. The user can then enter the appropriate information in the text boxes, select the appropriate strength model, and click on **Continue** to open the next data entry window.

It is often most efficient to define a new set of material properties by editing a previously defined set of properties which act as defaults. Defaults for the general data window (Fig. 6.4) can be obtained from the MATERIAL.TXT data file by clicking on the button **Select Data From MATERIAL.TXT to Act As Defaults** at the bottom of this window. This command button causes the select material to be used for default properties

window (Fig. 6.5) to be displayed. Clicking the **Continue** button of the window of Fig. 6.5 will cause the properties of the selected material to be inserted as default values into the text boxes of Fig. 6.4. This default insertion capability is also available for the following four material data input windows which will now be discussed.

After clicking the **Continue** button of the general material data window (Fig. 6.4) then the appropriate strength model constant data input window appears. Fig. 6.6 shows the Johnson-Cook model strength constant input window. The other strength models have similar input windows that reflect the constants required for those models. Clicking the **Continue** button of the strength model constant data input window (Fig. 6.6) causes the equation of state and artificial viscosity window to appear, Fig. 6.7. Again clicking the **Continue** button will open the fracture model constants window, Fig. 6.8. Clicking the **Continue** button of this window will cause the last material constants data input window (thermal Constants) to appear, Fig. 6.9. Finally, the user will be given the option of storing the just input material constants in the data file MATERIAL.TXT for use by TAYLOR immediately and for later reference.

As described above, TAYLOR can be used to add sets of material property coefficients to the data file MATERIAL.TXT. Data sets can also be deleted from MATERIAL.TXT. Clicking on the **Manage Custom Materials** button at the bottom-left corner of the main program window (Fig. 6.1) will cause the window of Fig. 6.3 to be displayed as was described previously. Selecting the **Delete Existing Custom Material Properties** option of Fig. 6.3 and then clicking **Continue** will cause a listing of the materials currently in MATERIAL.TXT to be displayed. The material to be deleted is selected by clicking on (and highlighting) the appropriate name in the listing and then clicking **Continue**. The user will then be asked to confirm the decision to delete.

The desired impact velocity is entered in the appropriate text box on the left-center side of the main program window, Fig. 6.1. The central portion of the main program window provides text boxes for entering the desired duration of the simulation and the data output interval (to data files NODEDATA.TXT and ELEMDATA.TXT, see Section 4). Selecting a relatively small data output interval will result in a fine time resolution of calculated results for subsequent postprocessing but the EPIC solution time will be increased and the output data files can become very large.

The final step for the preprocessor is to write out an EPIC input deck by clicking the **Write Input Deck** button at the bottom of the main program window, Fig. 6.1. Clicking the **Run Code** button beside this button will cause EPIC to perform a set of calculations based on the current input deck as described in the next section.

Note that previously calculated results stored in data files NODEDATA.TXT and ELEMDATA.TXT are deleted before a new EPIC run is initiated. Thus, these files, along with the EPIC input deck that was used to generate them, should be renamed (or moved to another directory) before launching EPIC in order to preserve the results. Renaming these three files back to their original names will allow the previously generated results to be postprocessed again.

7. How the TAYLOR Program Launches EPIC

EPIC was compiled using Microsoft FORTRAN PowerStation which produces a DOS executable. Since EPIC is such a large code special provisions must be made to run the code without leaving the Windows shell. This was accomplished by using a Windows program interface file (.PIF). The TAYLOR program launches the .PIF file (using the Visual BASIC Shell command) which in turn launches the EPIC executable. EPIC takes over the computer while it runs and then returns control to the TAYLOR program on completion of the calculations. This whole process is handled seamlessly for the user by TAYLOR when the **Run Code** button (Fig. 6.1) is clicked. Problems associated with running TAYLOR under Windows 95 (which treats DOS differently) were discussed in Section 5.2.

8. The Postprocessor of the TAYLOR Program

TAYLOR can display the last set of results calculated by EPIC in three basic ways, as indicated by the three **Select Output** zones along the right side of the main program window, Fig. 6.1. Displacements can be animated, data can be plotted as a function of time, and data can be displayed as colored and gray-scale contours.

Selecting **Displacement Animation** (top-right) and clicking the **Run Post Processor** button (bottom-central) of the main program window (Fig. 6.1) will pop open the displacement animation display window, Fig. 8.1. A typical deformed specimen (with very coarse mesh) is shown in Fig 8.1. Clicking the animate button will cause an animation of the deforming specimen to appear. The speed and fidelity of the animation can be controlled by changing the **Number of Cycles for Animation Display** (bottom-center Fig. 8.1) from the default value of 10. Clicking the **Plot Experimental Profile** (bottom-left Fig. 8.1) will cause the experimentally measured (or otherwise obtained) Taylor specimen profile contained in file PROFILE.TXT to be superimposed on the animation display. The ASCII PROFILE.TXT file consists of lines of data having the format (axial coord., radial coord.) starting from the impacted end. The first line of PROFILE.TXT is reserved for comments. Clicking the **Write Calculated Profile** will cause TAYLOR to write an ASCII data file (same format as PROFILE.TXT), of a user specified name, at a user specified time after impact, of the calculated specimen deformed profile. Note that the elapsed time after impact is displayed as the displacement animation proceeds. Clicking the **Print** button of Fig. 8.1 will cause the deformed geometry (for the elapsed time when the **Print** button was clicked) to be printed on the printer connected to the computer. Clicking the **Exit** button of Fig. 8.1 will cause control to return to the main program window, Fig. 6.1.

Selecting the **Z Velocity Plots** (or any other **Plots** options) of the main program window (Fig. 6.1) will allow the user to select (by clicking on with the cross hair cursor) as many as three nodes from the specimen mesh display box. The selected nodes are indicated by color coded circles, Fig. 8.2. For guidance while selecting the nodes, the R and Z coordinates of the cross hair cursor are shown in small boxes in the upper-right corner of the specimen mesh display box, Fig. 8.2. Clicking the **Run Post Processor** button (see Figs. 6.1 or 8.2) will cause the plot display window (in this case for Z velocity) to pop open, Fig. 8.3. Plots of the selected variable (Z velocity,

pressure, effective stress, effective strain, log strain rate, damage fraction, and temperature) at the selected nodal positions are plotted in the plot window as a function of elapsed time from impact. The initial positions of the various nodes are indicated at the top of the plot. The smaller the **Data Output Interval** (Fig. 6.1) the higher the fidelity of the plots. Note that the rezoning procedure of EPIC, where nodal positions are adjusted to minimize mesh distortions, adds a slight error to the plots. A hard copy of the plot can be obtained by clicking the **Print** button of Fig. 8.3. The plot colors will be maintained if a color printer is used. Clicking the **Exit** button of Fig. 8.3 returns the user to the main program window of TAYLOR, Fig. 6.1.

The plotted data can also be viewed in the form of contour plot animations. This is accomplished by first selecting the data to be shown as contours using the option buttons in the bottom right corner of the main program window, Fig. 6.1. Having clicked the data option button then the user clicks the **Run Post Processor** button and contour animation display window will appear, Fig. 8.4. The user has three options available for controlling the contour animation display. As was the case for displacement animation, the **Number of Animation Cycles** for the display can be set (see the bottom of Fig. 8.4, 25 is the default). The contours are displayed by breaking the specimen down into a number of rectangles which are assigned colors or gray-scale shades depending on interpolated function values. The number of rectangles used is controlled by the **Resolution Index** (see the bottom of Fig. 8.4, 10 is the default). A high **Resolution Index** produces a finer contour plot but requires more time for interpolation calculations. Finally, the **Gray Scale Contours** option (bottom-left, Fig. 8.4) can be selected where instead of color contours, high resolution gray-scale contours are shown instead. The gray-scale contours look best with the video card of the computer set to at least 32K colors, although in this video mode hard copies can not be produced. In the 256 color mode, hard copies of the contours can be produced by clicking the **Print** button. Figure 8.5 shows typical color contour strain rate data. Clicking the **Exit** button of Fig. 8.5 returns the user to the main program window of TAYLOR, Fig. 6.1.

9. The Material Strength Model Coefficient Optimizer of the TAYLOR Program

Allen (1995) proposed a technique for optimizing strength model coefficients to minimize the volume difference between calculated and measured Taylor specimen deformed geometries. The volume difference is composed of the sum of two components: the longitudinal volume difference and the radial volume difference. The longitudinal volume difference is defined as the magnitude of the difference in length between the measured and calculated specimen geometries times the undeformed specimen cross sectional area. The radial volume difference is obtained by first scaling the profiles to be the same length, slicing (in a numerical sense) the specimens into a number of disk-like segments, calculating the magnitude of the volume difference between the measured and calculated disk geometries, and then summing up all these radial volume errors. Note magnitudes are used so all errors are considered positive or additive. Thus, using an optimizer to minimize the volume difference will tend to force both the length and shape of the numerically calculated (with EPIC) deformed geometry to approach that of the measured. For convenience, the optimizer works with the nondimensional **% Relative Volume Error** which is

defined as the volume error (as discussed above) divided by the volume of the deformed test specimen, times 100. The optimizer attempts to minimize the **% Relative Volume Error** by adjusting the magnitudes of the strength model coefficients.

Allen used a response surface approach to optimization which required the intervention of a very knowledgeable user. Here, Allen's approach was extended and made completely seamless so no user intervention is required and the entire optimization process is automatic. Instead of using a response surface the well known conjugate direction method (Vanderplaats, 1984) was applied. This a gradient vector based, first order, search technique.

The optimizer is set up to work with all the material strength models contained in EPIC. An initial set of reasonable strength model coefficients, and the experimentally measured Taylor specimen deformed profile (data placed in file PROFILE.TXT) are required before the optimizer can be run.

In addition, the optimizer can also accommodate measured yield stress data from other tests such as the split Hopkinson bar or conventional quasi-static tension test data. This auxiliary data must be entered in an ASCII data file called YLD_STRS.TXT with data records having the format:

{	<i>⇒ start of record indicator</i>
Quasi-Static Yield Test	<i>⇒ text describing data source</i>
4.443E4	<i>⇒ measured yield stress</i>
0.002	<i>⇒ effective plastic strain</i>
1.667E-4	<i>⇒ effective plastic strain rate</i>
0	<i>⇒ homologous temperature</i>
}	<i>⇒ end of record indicator</i>

Any number of data records may be included in YLD_STRS.TXT. A nondimensional percentage error is calculated for each auxiliary data record as follows:

$$\text{Error} = \text{Abs} \left(\frac{\sigma_{ys}^P - \sigma_{ys}^M}{\sigma_{ys}^M} \right) \times 100 \quad (9.1)$$

where σ_{ys}^P is the yield stress calculated from the current values of the strength model coefficients (which vary as the optimization process proceeds) and σ_{ys}^M is the measured yield stress contained in the data file YLD_STRS.TXT. "Abs" indicates absolute value in equation (9.1). An error is calculated using equation (9.1) for each data record in YLD_STRS.TXT and these errors are added to the **% Relative Volume Error** discussed

previously to produce the total error that the optimizer seeks to minimize. This auxiliary data can not be used with the Bodner-Partom or MTS strength models as they are too complex for simple evaluation in equation (9.1).

Ideally, the optimizer would adjust the strength model coefficients until a set of coefficients were found that produced a calculated Taylor deformed specimen geometry identical to that of the measured and the calculated yield stresses would exactly agree with the measured yield stresses (from YLD_STRS.TXT). However, due to various types of errors this ideal fit to the data will not occur. In reality, the optimizer will find a compromise solution where the errors with respect to all the experimental data used will be reduced to approximately the same level.

The optimizer is started by clicking the **Optimize Constants** button the bottom-right side of the main program window, Fig. 6.1. The user will then be prompted for optimization parameters: **Number of Iterations** and **Search Distance Factor**, Fig. 9.1. The optimizer improves the strength model constants through a series of iterations. The accuracy of the strength model coefficients improves as more iterations are used but at the expense of additional computational time. Each iteration requires one EPIC solve per strength model coefficient (for gradient determination) and three additional EPIC solves for conducting the one dimensional search through the design space as required by the conjugate direction method. Thus, the Johnson-Cook strength model requires eight EPIC solves per iteration, which can be very time consuming for models with fine finite element meshes. It is best to select a relatively low number of iterations (say 3) and then rerun the optimizer to further refine the coefficients if the initial optimization results look promising. The optimizer can be rerun as many times as desired.

The other required optimizer parameter is the **Search Distance Factor** which is related to the maximum allowable percentage change in the strength model coefficients in one iteration. A large **Search Distance Factor** (say 0.2) often allows the optimizer to reach the vicinity of the best set strength model coefficients with few iterations but fine tuning to the absolute best possible set of coefficients may not be possible. Thus, it is often a good idea to use a large **Search Distance Factor** for an initial optimizer run and then a small (say 0.05) **Search Distance Factor** for a second optimization run. Clicking the **Continue** button closes the optimizer parameters window.

Next the user is prompted (see Fig. 9.2) for a new name for the optimized material to avoid confusion with the original set of material properties which may be stored in the data file MATERIAL.TXT. A default value for the new material name is given in the text box (Fig. 9.2) which consists of the old name plus the current date and time. This is a convenient way to keep track of multiple sets of optimization run results. Finally, the user will be asked if the auxiliary experimental data contained in YLD_STRS.TXT (as described above) should be used by the optimizer. While the optimizer runs messages like “*Iteration: 1 Gradient calculation 1 of 5*” are displayed on the screen to constantly inform the user about the status of the optimization calculations. The user will be given the option of adding the new, optimized set of material coefficients to data file MATERIAL.TXT on completion of the optimization process.

A simple numerical test of the optimizer with simulated experimental data was conducted as follows. First a typical OFHC copper Taylor specimen was numerically modeled. The displacement animator was used to write out a file (**Write Calculated Profile** button, Fig. 8.1) containing the final deformed shape (120 μ s after impact) of the specimen. This file was named PROFILE.TXT and was used as a convenient means to simulate a set of experimental Taylor test results. The impact velocity was then increased by 10% and the EPIC analysis rerun with all other model parameters unchanged. The increased velocity produced an artificial discrepancy between the artificial "measured" results in the PROFILE.TXT file and the results of the higher velocity EPIC run. This discrepancy is shown in Fig. 9.3 where (as expected) the "measured" profile (slightly thicker and lighter line) exhibits less mushrooming and is longer than the calculated profile (narrower, dark lines that also show element boundaries).

The optimizer was then used to adjust the strength model constants to force the higher velocity numerical model to match up with the "measured" results (the lower velocity model). These results are shown in the material property optimization display window, Fig. 9.4. This window is displayed for a few seconds after each iteration is completed and stays open until closed by the user (using the **Exit** button) after all iterations have been completed. The left side of the window displays a box containing the deformed shape of the Taylor specimen as calculated by EPIC using the optimized material strength coefficients. This box also displays the measured specimen profile (contained in PROFILE.TXT) for comparison. As can be seen in Fig. 9.4, the calculated and "measured" profiles for this simple, simulated test case are virtually identical. This means that for this artificial test case the optimizer was able to modify the strength model coefficients in some fashion to compensate for the 10% velocity increase.

On the right side of Fig. 9.4 plots of **Objective Function** and % **Relative Volume Error** versus optimizer iteration number are shown. The **Objective Function** is defined as the % **Relative Volume Error** plus terms (of the form of equation (9.1)) from the auxiliary experimental data contained in file YLD_STRS.TXT. For the simple, simulated test case shown in Fig. 9.4, auxiliary data was not used so both plots are identical. Note that the plots of Fig. 9.4 show typical optimization results - a great improvement after one iteration and then much smaller improvements for subsequent iterations.

The optimizer was applied to a real OFHC copper Taylor test data set, Fig. 9.5. Figure 9.5(a) indicates that nominal OFHC copper strength model coefficients did a relatively poor job of representing the actual specimen behavior. Figure 9.5(b) shows that an almost perfect match between measured and calculated Taylor specimen profiles was obtained using the optimized strength model coefficients.

10. Suggestions for Future Work

The following suggestions for future work are offered:

- Make the TAYLOR program and associated EPIC code fully compatible with Windows 95.
- Investigate, in some detail with real data, the effectiveness of optimizing the material strength model coefficients (for Johnson-Cook, Zerilli-Armstrong, Bodner-Partom, MTS models) simultaneously using

multiple sets of data (Taylor test, Hopkinson pressure bar, quasi-static tension test). Using multiple independent sets of data from different strain and strain rate regimes should increase the reliability of the strength model coefficients.

- Modify the TAYLOR program to conduct material strength model coefficient optimization runs simultaneously using data from more than one Taylor test (multiple specimens of different size, shape, and impact velocity). Using multiple independent data sets for the fitting process should lead to more reliable strength model coefficients.
- Modify the TAYLOR program to numerically model the behavior of the target as well. Currently, TAYLOR treats the target as a numerically rigid interface as do most researchers.
- Develop a program (similar to TAYLOR) for analyzing and optimizing shaped charges (SC) and explosively formed penetrators (EFP). This program would allow designers (with little knowledge of EPIC or computational mechanics) to design, analyze, and optimize SCs and EFPs in a seamless and user-friendly computational environment.
- Investigate the possibility of making the special compact nodal and elemental data output files developed especially for the TAYLOR program (PAT = 6 case) a permanent part of the mainstream EPIC code. These neutral files will provide easy access to EPIC output for those users without PATRAN.

11. Acknowledgments

The support provided by the AFOSR Summer Faculty Research Extension Program made this project possible. Contract monitors Joseph C. Foster, Jr. and Joel W. House, both of Wright Lab., Eglin AFB, provided the writer with a framework for understanding the technical issues associated with this project. This is much appreciated. William H. Cook (Wright Lab., Eglin AFB), and John A. Collins (formerly of Wright Lab., Eglin AFB), introduced the writer to EPIC, for which the writer is very grateful. John A. Collins especially encouraged the application of Visual BASIC to the task of making EPIC more user friendly for those not working in the area of computational mechanics. Gordon R. Johnson, of Alliant Techsystems Inc., graciously provided the writer with the source code for Research EPIC 1995, thus providing free access to a monumental investment in time, money, and effort. David J. Allen, formerly a graduate student (U of Alabama - Tuscaloosa) of the highest quality and now a mountain bike racer, did all the hard work to coax EPIC to compile under Microsoft FORTRAN PowerStation, and led the way for using EPIC to optimize material strength model coefficients. Stanley E. Jones (U. of Alabama - Tuscaloosa) furnished Taylor test data for use while developing the TAYLOR program, and also provided much insight into high strain rate phenomena through many technical discussions. Administrative support for this project was provided by Scott Licoscos of Research and Development Laboratories.

12. References

Allen, D. J., "Use of the Taylor Impact Test to Determine Constants for Material Strength Models," MS Thesis, Dept. of Engineering Science and Mechanics, University of Alabama, Tuscaloosa, 1995.

Bodner, S. R., and Merzer, A., "Viscoplastic Constitutive Equations for Copper with Strain Rate History and Temperature Effects," *Journal of Engineering Materials and Technology*, 100, October, 1978.

Bodner, S. R., and Partom, Y., "Constitutive Equations for Elastic-Viscoplastic Strain-Hardening Materials," *Journal of Applied Mechanics*, June, 1975.

Cook, W. H., Rajendran, A. M., and Grove, D. J., "An Efficient Numerical Implementation of the Bodner-Partom Model in the EPIC-2 Code," *Engineering Fracture Mechanics*, 41, 1992.

Follansbee, P. S., and Gray, G. T., III, "An Analysis of the Low Temperature, Low and High Strain Rate Deformation of Ti-6AL-4V," *Met. Trans.*, 20A, p. 863-874, 1989.

Follansbee, P. S., and Kocks, U. F., "A Constitutive Description of the Deformation of Copper Based on the Use of Mechanical Threshold Stress as an Internal State Variable," *Acta Metall.*, 36(1), p. 81-93, 1988.

Hawkyard, J. B., "A Theory for the Mushrooming of Flat-Ended Projectiles Impinging on a Flat Rigid Anvil, Using Energy Considerations," *Int. J. Mech. Sci.*, 11, p. 313, 1969.

Holmquist, T. J., and Johnson, G. R., "Determination of Constants and Comparison of Results for Various Constitutive Models," *Journal de Physique IV*, October, 1991.

House, J. W. "Taylor Impact Testing," Report No. AFATL-TR-89-41, Air Force Armament Laboratory, Eglin AFB, FL, September, 1989.

Johnson, G. R., "EPIC-3, A Computer Program for Elastic-Plastic Impact Calculations in 3 Dimensions," Final Report, Honeywell Inc., Defense Systems Division, May, 1977.

Johnson, G. R., "Material Characterization for Warhead Computations," in *Tactical Missile Warheads*, J. Carleone Editor, Vol. 155 *Progress in Astronautics and Aeronautics*, AIAA, 1993.

Johnson, G. R., "Linking of Lagrangian Particle Methods to Standard Finite Element Methods for High Velocity Impact Computations," *Nuclear Engineering and Design*, 150, p. 265-274, 1994.

Johnson, G. R., and Cook, W. H., "A Constitutive Model and Data for Metals Subjected to Large Strains, High Strain Rates, and High Temperatures," 7th International Symposium on Ballistics, The Hague, The Netherlands, April, 1983.

Johnson, G. R., and Cook, W. H., "Fracture Characteristics of Three Metals Subjected to Various Strains, Strain Rates, Temperatures and Pressures," *Engineering Fracture Mechanics*, 21(1), p. 31-48, 1985.

Jones, S. E., Gillis, P. P., and Foster, J. C., Jr., "On the Equation of Motion of the Undeformed Section of a Taylor Impact Specimen," *J. of Applied Phys.*, 61, p. 499, 1987.

Karpp, R. R., "Warhead Simulation Techniques: Hydrocodes," in *Tactical Missile Warheads*, J. Carleone Editor, Vol. 155 *Progress in Astronautics and Aeronautics*, AIAA, 1993.

Lee, E. H., and Tupper, S. J., "Analysis of Plastic Deformation In a Steel Cylinder Striking a Rigid Target," *J. of Applied Mech.*, 31, p. 63-70, 1954.

Rakhmatulin, K. A., "Propagation of Wave on Unloading," *Applied Mathematics and Mechanics*, Leningrad, 19, p. 91, 1945.

Taylor, G. I., "The Use of Flat-Ended Projectiles for Determining Dynamic Yield Stress," *Proc. Roy. Soc. London, Series A*, Vol. 194, p. 289, 1948.

Vanderplaats, G. N., "Numerical Optimization Techniques for Engineering Design with Applications," McGraw-Hill Book Company, New York, 1984

Zerilli, F. J., and Armstrong, R. W., "Dislocation-Mechanics-Based Constitutive Relations for Material Dynamics Calculation," *Journal of Applied Physics*, 61, 1987.

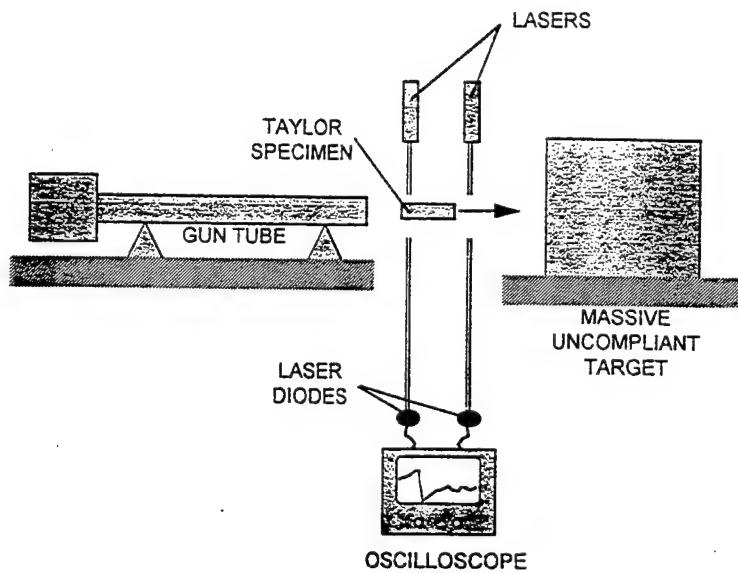


Fig. 2.1 Schematic Drawing of Taylor Test Equipment

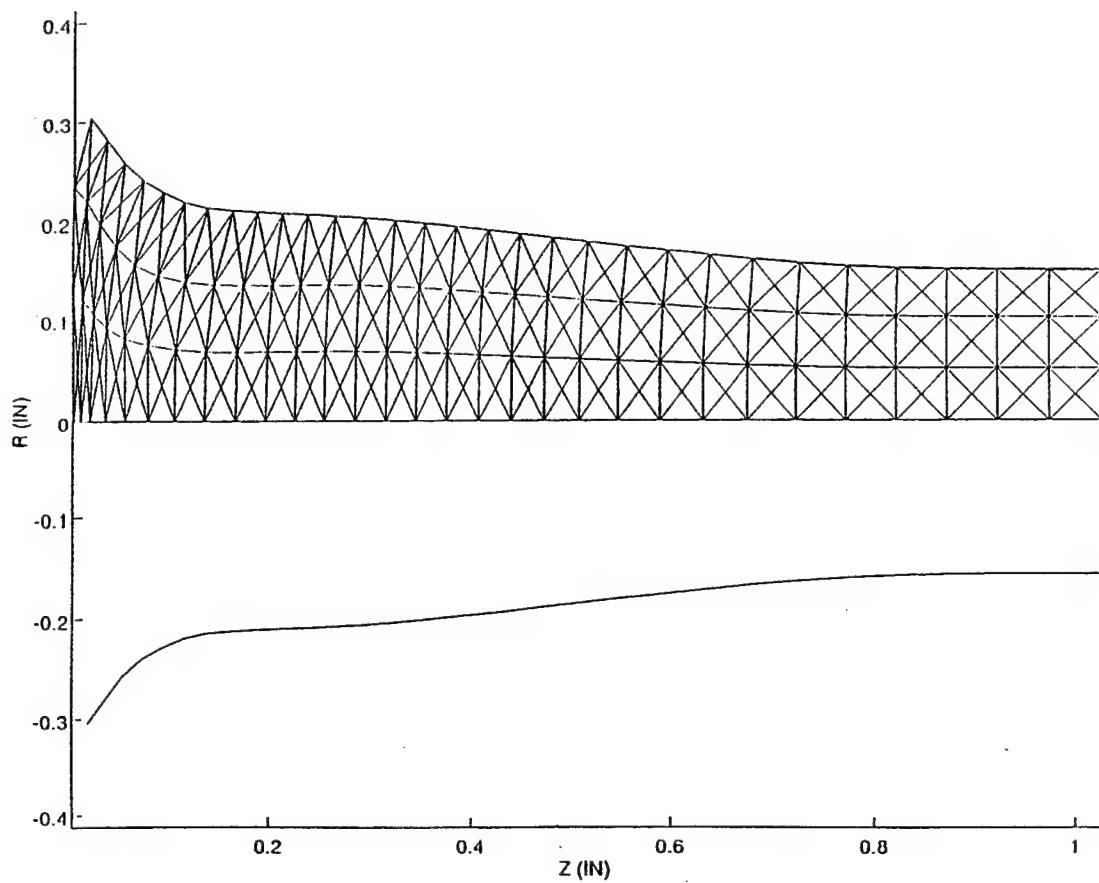


Fig. 3.1 Typical Deformed Mesh of Three-Noded, Triangular, Axisymmetric Elements

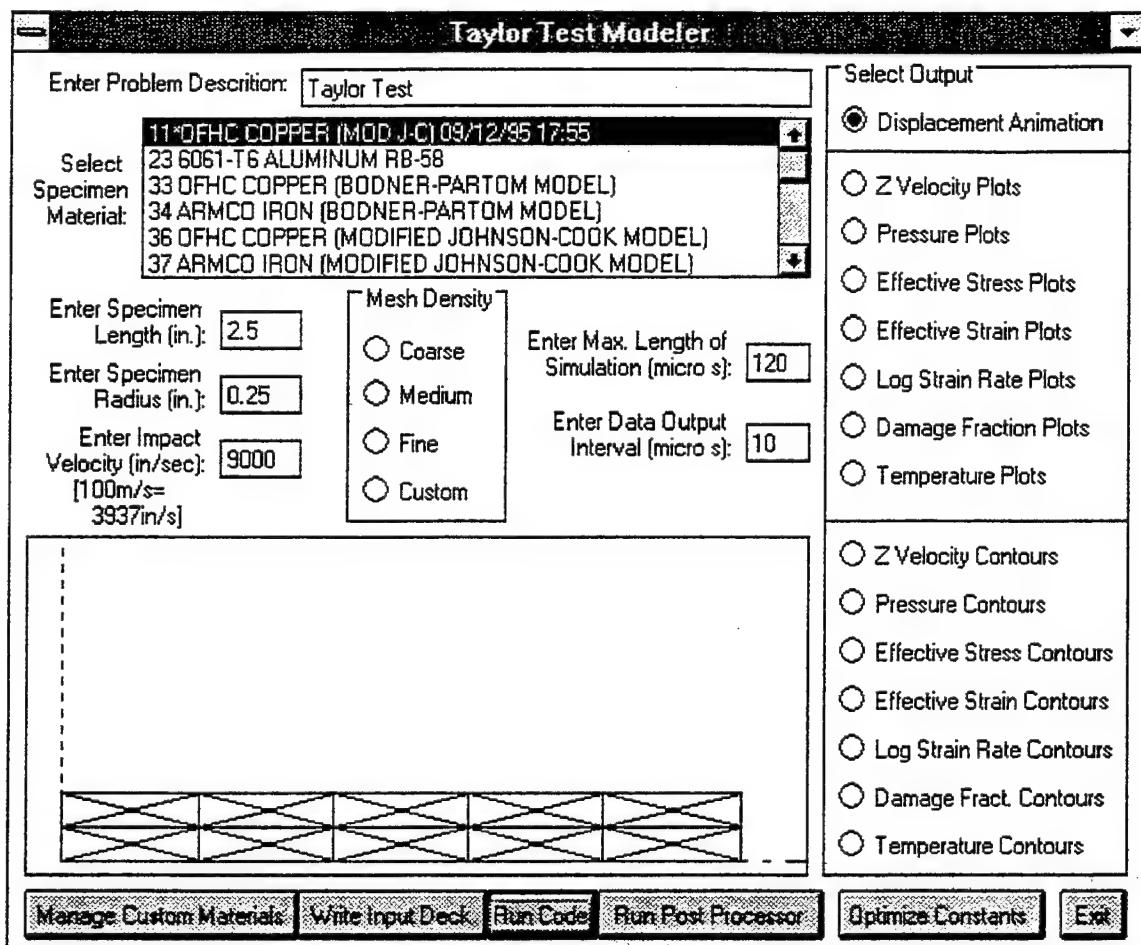


Fig. 6.1 Main TAYLOR Program Window

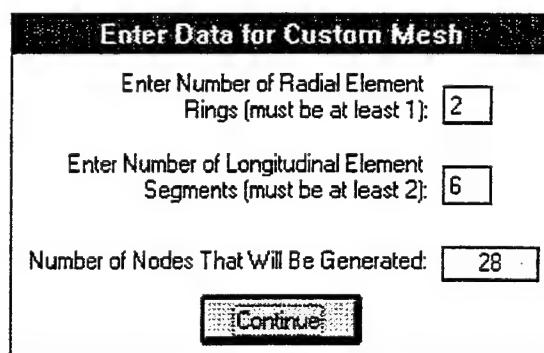


Fig. 6.2 Custom Mesh Definition Window

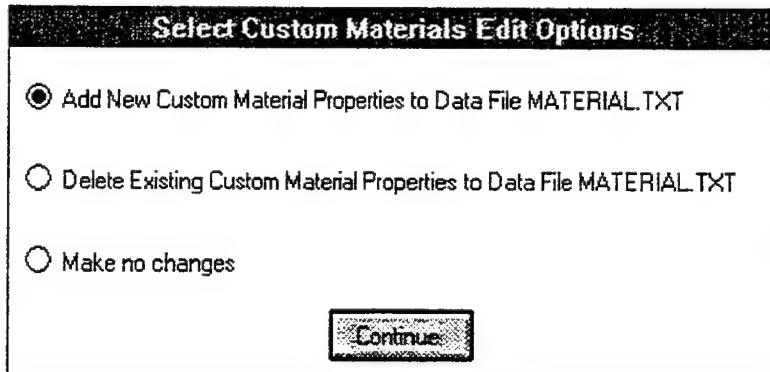


Fig. 6.3 Manage Custom Materials Main Window

The dialog box has a title bar 'Enter General Material Data'. It contains input fields for 'Material Description', 'Density ([lb/in^3]/(386.4 in/s^2))', 'Specific Heat ([in.lb]/([lb.degF])*(386.4 in/s^2))', 'Specimen Temperature (degF)', 'Ambient Temperature (degF)', 'Melting Temperature (degF)', and 'Abs. Zero Temperature (degF)'. Below these is a list titled 'Select Strength Model' with options: 1 Johnson-Cook Strength Model, 2 Modified Johnson-Cook Strength Model, 3 Zerilli-Armstrong FCC Strength Model, 4 Zerilli-Armstrong BCC Strength Model, 5 Bodner-Partom Strength Model, and 6 MTS Strength Model. The second option is selected. At the bottom are buttons 'Select Data From MATERIAL.TXT to Act As Defaults' and 'Continue'.

Fig. 6.4 General Material Data Window

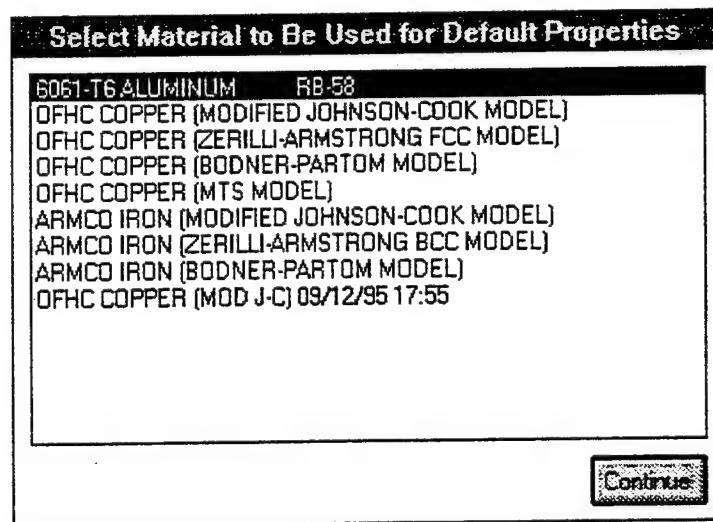


Fig. 6.5 Select Material to be Used for Default Properties Window

A screenshot of a software dialog box titled "Enter Johnson-Cook Model Strength Constants". It contains fields for inputting strength constants:

Shear Modulus (psi):	<input type="text"/>
C1 (psi):	<input type="text"/>
C2 (psi):	<input type="text"/>
N (dimensionless):	<input type="text"/>
C3 (dimensionless):	<input type="text"/>
M (dimensionless):	<input type="text"/>
C4 (dimensionless):	<input type="text"/>
SMAX (psi):	<input type="text"/>

At the bottom are two buttons: "Select Data From MATERIAL.TXT to Act As Defaults" and "Continue".

Fig. 6.6 Johnson-Cook Model Strength Constant Data Input Window

Enter EOS, PMIN, CL, CQ, CH Constants

K1 (psi):	<input type="text"/>
K2 (psi):	<input type="text"/>
K3 (psi):	<input type="text"/>
GAMMA (dimensionless):	<input type="text"/>
PMIN (psi):	<input type="text"/>
CL (dimensionless):	<input type="text"/>
CQ (dimensionless):	<input type="text"/>
CH (dimensionless):	<input type="text"/>

Select Data From MATERIAL.TXT to Act As Defaults **Continue**

Fig. 6.7 Equation of State and Artificial Viscosity Data Input Window

Enter Johnson-Cook Fracture Model Constants

D1 (dimensionless):	<input type="text"/>
D2 (dimensionless):	<input type="text"/>
D3 (dimensionless):	<input type="text"/>
D4 (dimensionless):	<input type="text"/>
D5 (dimensionless):	<input type="text"/>
SPALL (psi):	<input type="text"/>
EFRMIN (dimensionless):	<input type="text"/>
SOFT (dimensionless):	<input type="text"/>

Select Data From MATERIAL.TXT to Act As Defaults **Continue**

Fig. 6.8 Fracture Model Constants Input Window

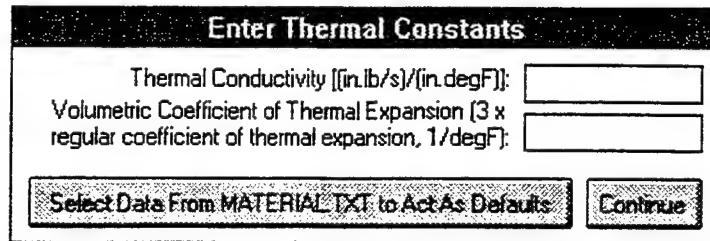


Fig. 6.9 Thermal Constants Data Input Window

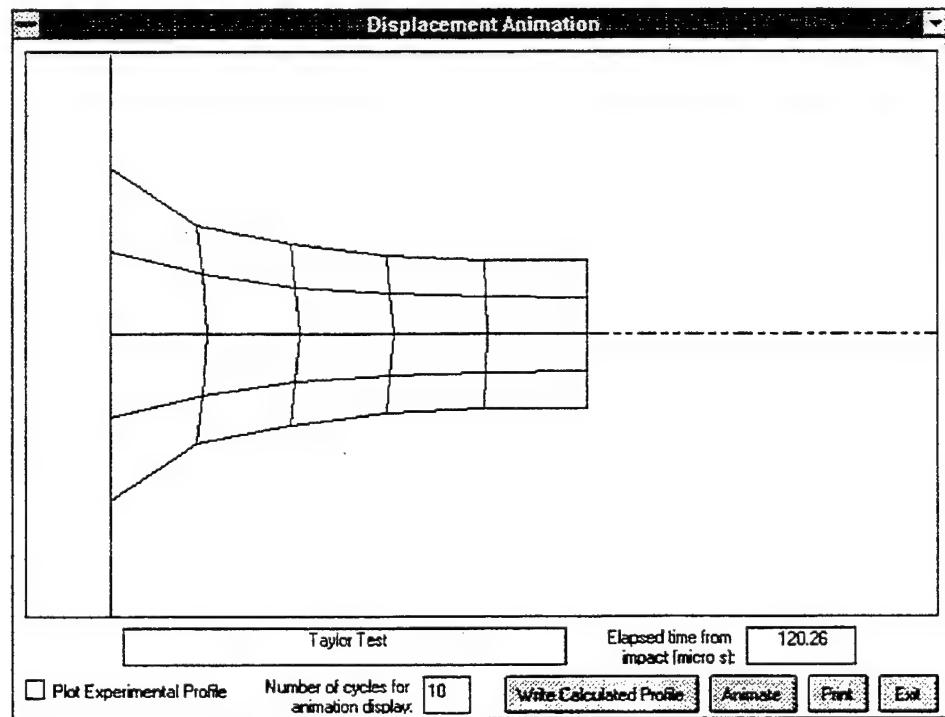


Fig. 8.1 Displacement Animation Display Window

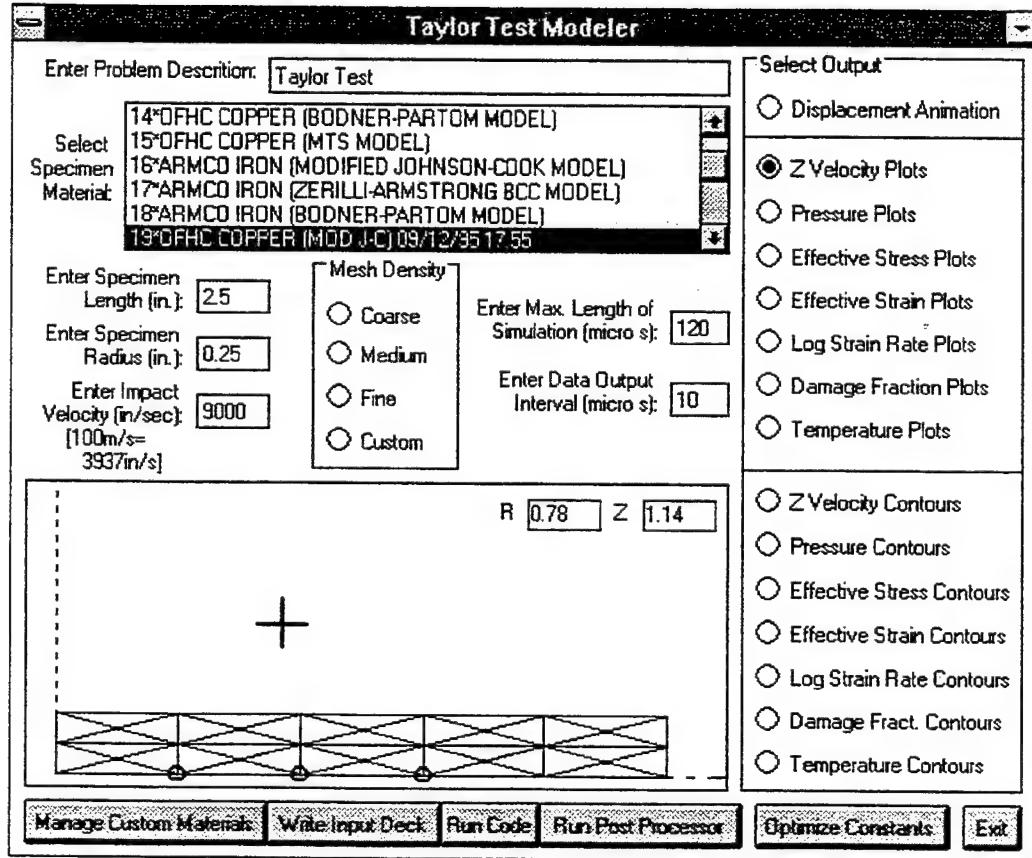


Fig. 8.2 Selecting Nodes for Plotting

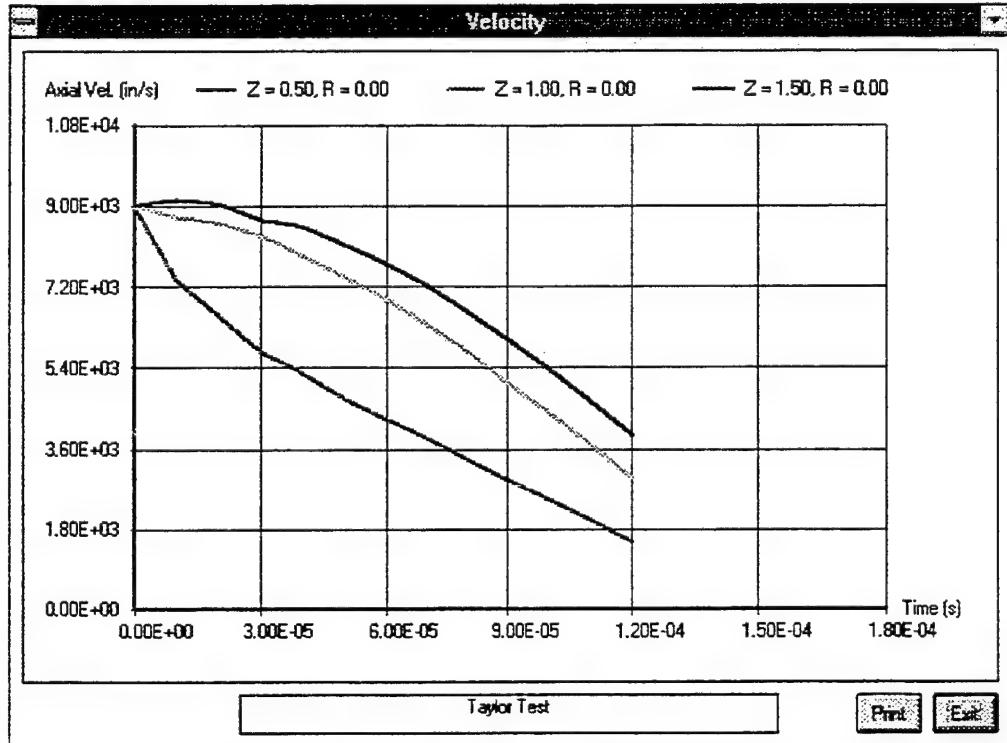


Fig. 8.3 Typical Plot Showing Z Velocity as a Function of Elapsed Time from Impact for Three Selected Nodes

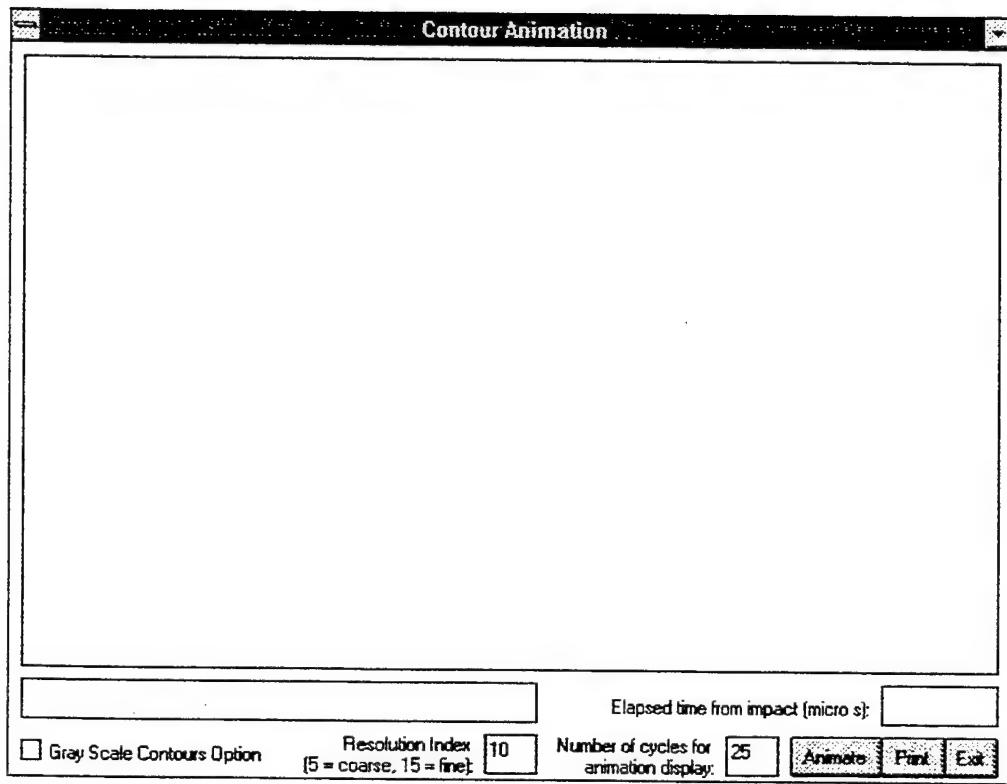


Fig. 8.4 Contour Animation Display Window

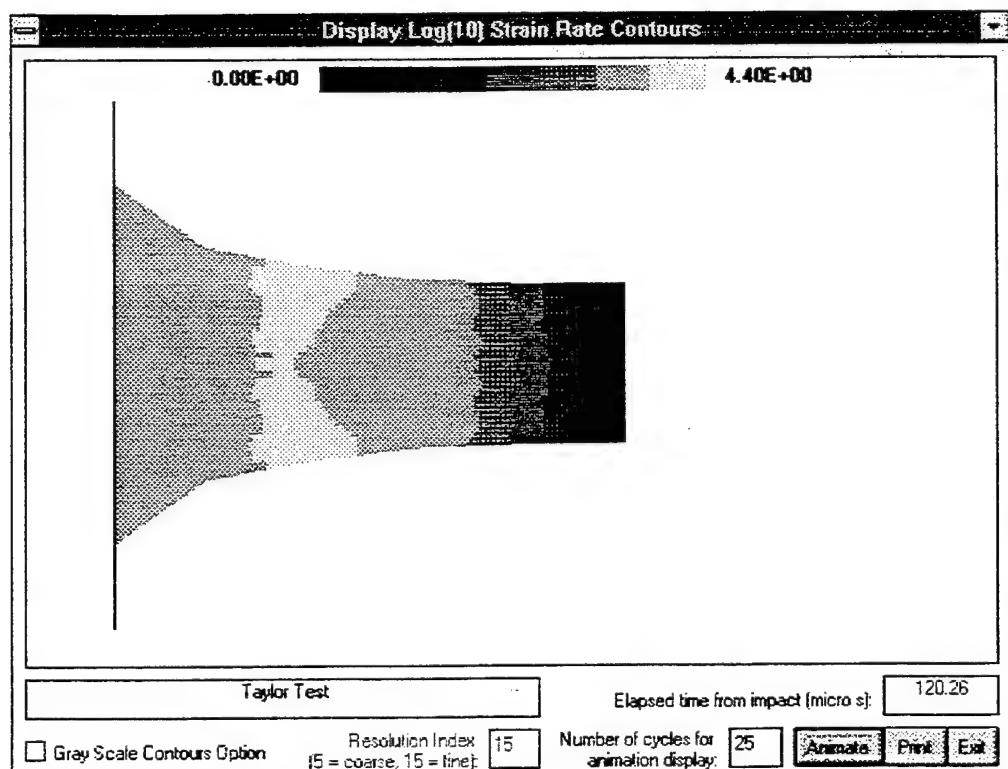


Fig. 8.5 Contour Animation Display Window Showing Typical Strain Rate Data

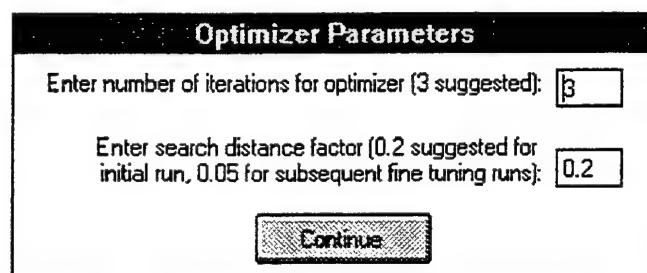


Fig. 9.1 Optimization Parameters Input Window

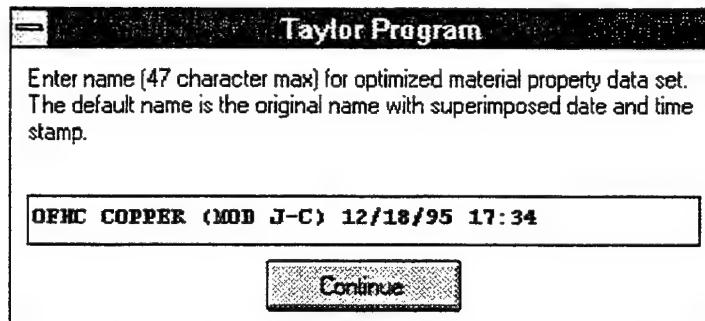


Fig. 9.2 New Material Name Input Window

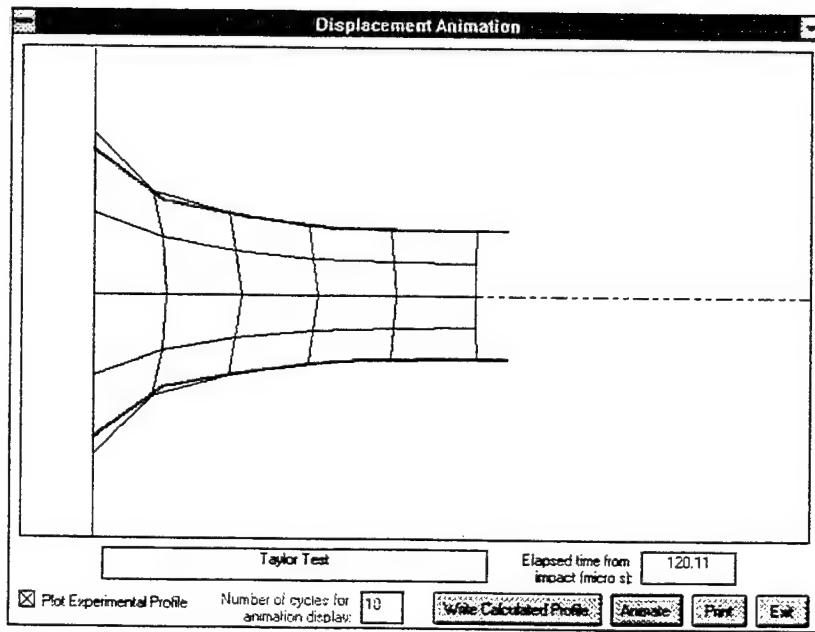


Fig. 9.3 Initial Mismatch Between Calculated and Measured Specimen Profiles for Simple Optimizer Test Case

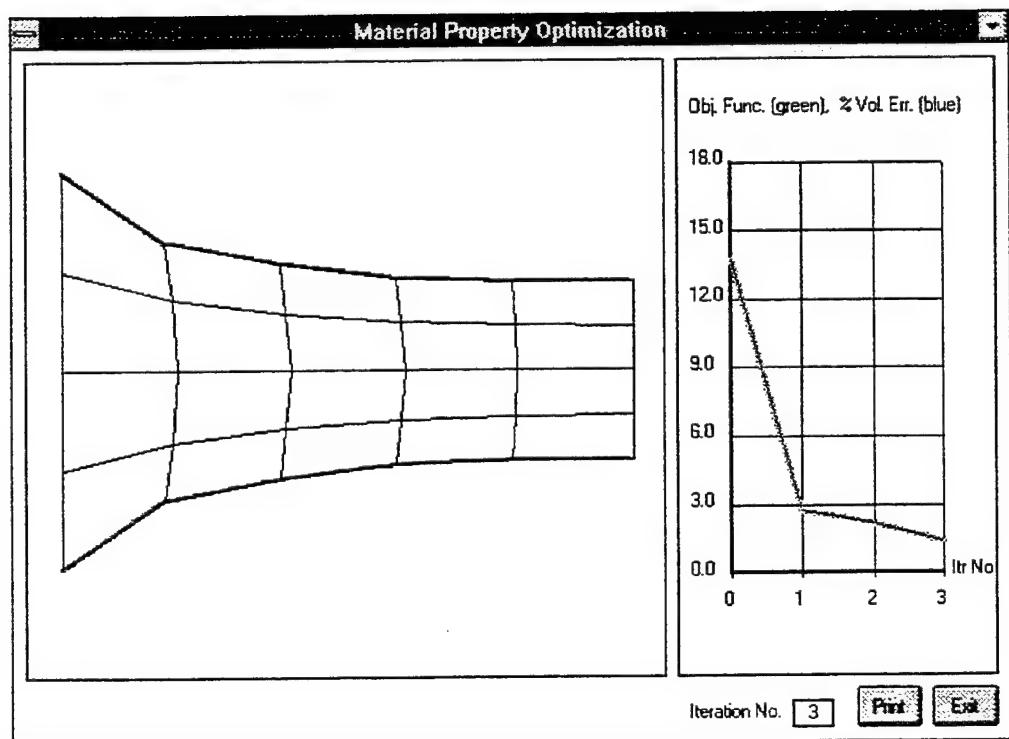
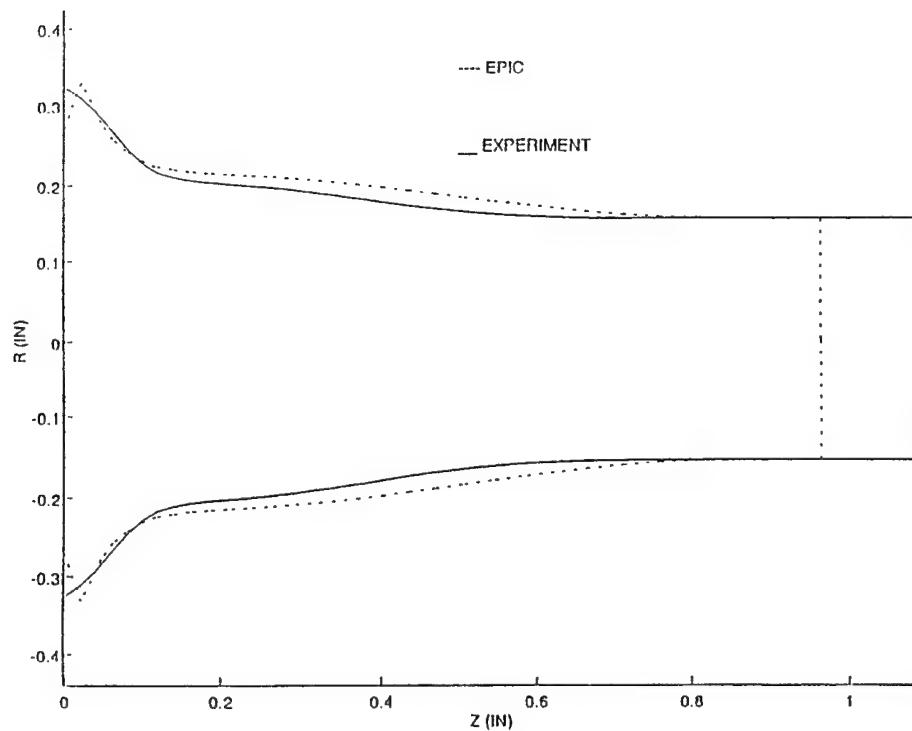
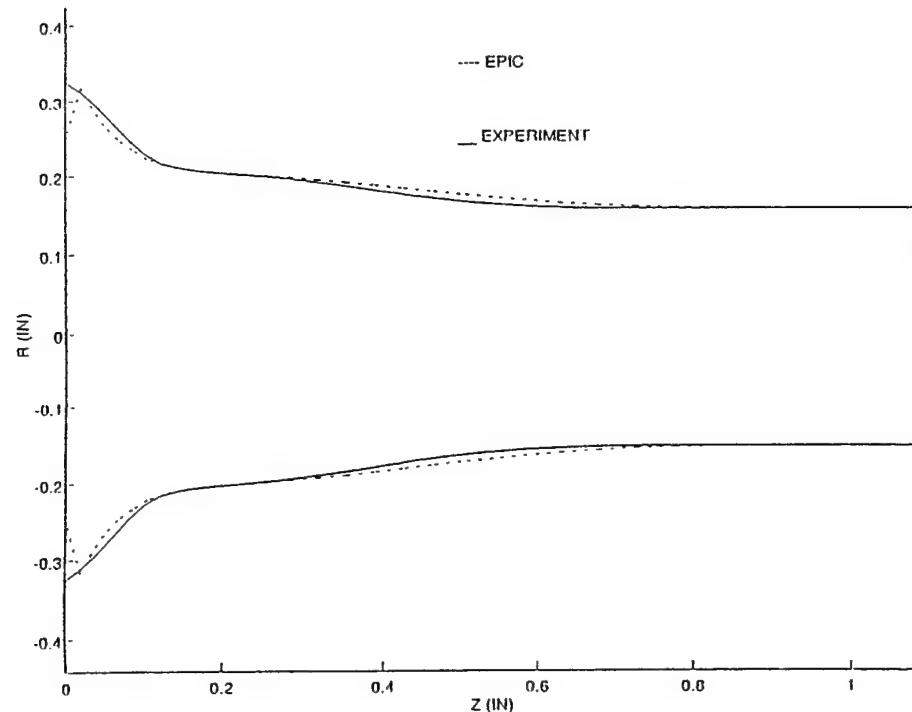


Fig. 9.4 Material Property Optimization Results Display Window for Simple Optimization Test Case



(a) Calculated and Measured Specimen Profile Before Optimization



(b) Calculated and Measured Specimen Profile After Optimization

Fig. 9.5 Material Property Optimization Results for OFHC Copper Specimen

John Schauer report unavailable at time of publication.

Neural Network Identification and Control in Metal Forging

Carla A. Schwartz
University of Florida

Department ECE
PO Box 116130
405 CSE, Building 42
Gainesville, FL 32611-6130

Final Report for:
Summer Faculty Research Program
Wright Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling AFB, Washington DC
and
Wright Laboratory

December, 1995

Neural Network Identification and Control in Metal Forging

Carla A. Schwartz
University of Florida
Dept of Electrical and Computer Engineering
405 CSE, Building 42
Gainesville, FL 32611-6130
email: carla@neptune.ee.ufl.edu

Abstract

Forged metal is used as a strengthening technique to form parts for a wide variety of applications. Precision forging is of interest to AFOSR in order to design engine parts, for example, with specified characteristics. Currently, many aspects of the forging process are imprecise and/or ad hoc. Thus, the metal forging area is ripe for the application of process control techniques to improve the forging process for efficient and effective manufacturing.

There have been some attempts to apply control techniques to forging in the past. Most of these involve using open loop control.

The success of a metal forging process may be measured by how satisfactorily final shape and final microstructural objectives are achieved. The final microstructure is often very sensitive to poorly known process parameters. Since some important processing signals are currently available for measurement, it is desirable to implement feedback control in order to compensate for this sensitivity. However, the task of designing a suitable controller is difficult, since there are no analytical process models which could be related to control variables which are measurable. Based on computations of thermo-mechanical properties of the forging process, in this report a neural network is used in the design of a direct controller for a simple extrusion of plain carbon steel. This is accomplished without first identifying a process model and then inverting it. Simulation results are provided.

This work is of note because of the novel use of the neural network to learn inverse dynamics for direct control. It is hoped that these techniques will be useful in many forging applications, as well as other control applications.

Neural Network Identification and Control in Metal Forging

Carla A. Schwartz

1. Introduction

In this work, the problem of implementing direct, real-time control of a metal forging process is addressed.

Forging is the process of altering the shape and mechanical properties of a piece of metal, (the *billet*), by deforming it through the action of one or more shaped *dies*. Nonlinear finite-element codes can predict the flow of the deforming metal with reasonable accuracy. Through painstaking experimental work, using semi-empirical modeling, the evolution of the crystal structure of the material can also be predicted, at least over a narrow range of temperatures and deformation rates. These large-scale and small-scale dynamics are coupled, since the microstructure strongly influences the bulk mechanical properties, and the velocity and temperature fields, which determine the microstructural dynamics, must be calculated across the entire workpiece simultaneously. Both the macroscopic and microscopic aspects of the process can be highly sensitive to process parameters that are poorly known, and difficult to measure. This makes forging a natural candidate for feedback techniques. However, sizable obstacles stand in the way of applying real-time feedback control. These obstacles include a shortage of controls, a shortage of measurements, and a lack of process models that capture the essential behavior of the system without requiring hours of computation.

Two important goals have motivated this work. The first is the need to incorporate microstructural models directly into the control of the large-scale process of forging. The second is the need for fast real-time models that capture the full behavior of the system, in some sufficiently broad operating region, (to be incorporated for controller design). The first goal is achieved by manipulating mathematical models of the microstructural dynamics. The

second is accomplished using an artificial neural network. These techniques are developed on a simple extrusion.

Although the eventual goal of this research is to provide closed-loop control of any forging, the general problem is quite complicated. This study focuses on controlling the simpler process of a round-to-round extrusion. In an extrusion, a ram forces a billet through a shaped die (in this case the aperture is circular). The analysis presented here is one-dimensional. Only the material behavior along the longitudinal axis of the billet (that is, the velocity vector of the ram) is considered, the radial variations are neglected. Furthermore, the length of the deformation zone in the extrusion die is assumed to be small compared to the length of the workpiece. Therefore, quantities such as strain rate and temperature are averaged inside the die. This approximation is helped by the choice of a constant strain rate shape for the die.

The control objective is to obtain a finished piece possessing a uniform microstructure along its extrusion axis. The microstructure is characterized by the average grain size. The relationship between recrystallized grain size and strain rate is temperature-dependent. The temperature of the material in the die will vary due to loss of heat to the container and die, and deformation heating. Thus the strain rate should vary as the temperature does. The control signal used to effect these changes is the ram velocity.

Using microstructural models, it is possible to write down a nonlinear feedback control law that gives the ram velocity as a function of certain billet temperatures. This would potentially preclude the necessity for the controller design. The problem is that these temperature values cannot be measured. On the other hand, the ram force represents a signal which is measurable. Using nonlinear finite element method (FEM) process models, it is possible to predict temperature and force profiles, based on given velocity profiles, volumes, shapes, materials models, and initial process parameters.

In this work, the FEM program is used as an input/output model to describe input/output

behavior of the forging as it relates velocity profiles, (the control), to ram force profiles (the measurement variable). For this application, standard techniques for designing model-based observers are not suitable, because the FEM codes may take hours to run one simulation of a forging process with a fixed set of parameter values, which is far too long for real-time control. It is also impossible to directly invert the algorithms. However, this does not preclude the possibility of using the FEM model as a process model for *off-line* controller design, as long as the controller outputs (velocities) can be computed quickly enough from the measurements (forces). This is precisely what was done in the work described in this report, wherein ANNs were trained directly from the FEM data as fast computational representations for the inverse of the input/output behavior of the FEM, for direct control design.

Recent work on isothermal forgings concentrates on the microstructural aspects (Berg *et al.*, 1994; Berg *et al.*, 1995b; Berg *et al.*, 1995a). Based on identified optimal processing conditions, these studies use an open-loop optimization technique to find the ram¹ velocity profile that best achieves those conditions. A similar approach to the nonisothermal problem is carried out in (Cheng *et al.*, 1994), wherein the initial die temperature and open-loop ram velocity profile are found which best achieve a desired strain rate and temperature history in a particular region of the billet. Tibbetts and Wen (Tibbetts and Wen, 1995) have used a reduced-order design model to find open-loop ram velocity profiles that minimize the power required in an isothermal extrusion. The current work attempts direct microstructural control over the entire billet for the nonisothermal case. The nonisothermal case is much more difficult than the isothermal case, since the microstructural behavior is very sensitive to temperature, and certain critical thermal process parameters are very poorly known. Due to this temperature sensitivity, the open-loop approaches of (Malas *et al.*, 1993; Berg *et al.*, 1995b; Berg *et al.*, 1995a; Cheng *et al.*, 1994; Tibbetts and Wen, 1995) are inadequate, in order to perform a satisfactory forging. The inadequacies of open loop control form a large part of the motivation for the use of a feedback control in a metal forging process described

¹ The ram is attached to and drives one of the dies as it stresses the billet to form it.

in this work.

Real-time, model-based closed loop control has also been proposed in (Meyer and Wadley, 1993) for hot isostatic pressing, using direct measurement of certain microstructural parameters. These measurements are currently unavailable for the forging problem, so a different approach is called for. Good microstructural evolution models are necessary for any practical approach to this problem. Such models are often proprietary, and are therefore scarce in the open literature. Devadas (Devadas *et al.*, 1991) surveys available results for austenitic plain carbon steel. This study considers the behavior of this material only during deformation. The dominant mechanism under typical hot working conditions is dynamic recrystallization. The dynamic recrystallization model for austenitic steel which is used in this study is provided in (Devadas *et al.*, 1991) and (Senuma and Yada, 1986).

There are two main contributions of this work. First, this work represents a novel approach to the control of metal forging processes using neural networks. Second, this work includes a novel approach to the use of neural networks in control of nonlinear systems, due to the direct controller design, without the necessity of identification of a forward plant model or the assumption of a parameterized control law.

This report is organized as follows. Section 2 briefly poses the extrusion as a control problem. Section 3 describes the microstructural and process models used to describe a simple forging. Section 4 is a brief introduction to the use of ANNs in control. Section 5 is a description of the use of an ANN in direct controller design for a forging process, including ANN training and simulation results. Section 6 contains some concluding remarks.

2. Forging as a Control Problem

In this section, the extrusion process is posed as a control problem. For the purpose of posing the extrusion as a control problem, the main objective of this work is to complete a round to

round hot extrusion of plain carbon steel with a uniform fine microstructure in the finished product. A uniform grain size is desirable to help assure uniform properties in the finished piece. As will be discussed in the next section, this microstructural objective can be described in terms of a uniform desired austenitic² grain size for design purposes. The mechanism considered, dynamic recrystallization, is active only during the actual deformation of the material, that is, only while the material is passing through the die.

The microstructural model for austenitic steel used here predicts grain size as a function of temperature and strain rate. One problem is that the thermal behavior of the billet can be complicated. The undeformed portion of the billet cools as it sits in the container prior to extrusion, which tends to lower the temperature of the deforming material. The material also heats up as it deforms, which, in the absence of any losses, will significantly raise the temperature of the deforming material. Therefore, the temperature profile in the die is typically not constant, so that applying a uniform ram velocity profile will not produce a uniform microstructure.

As mentioned in the introduction, in addition to having a desired objective to be achieved, (uniform average grain size), there is a measurement signal, (ram force), and a control signal, (ram velocity). The ram velocity is clearly a natural choice of input, and, (aside from the fact that force is the most readily measured signal), the ram force is a reasonable choice for the plant *output*, because it has a strong correlation with the temperature of the deforming material. If other signals, such as die temperatures, were available for measurement, they, too, could be included as outputs in the plant model.

Based on manipulation of the dynamic recrystallization model, in the sequel, a feedback relation which exists between the temperature of the material as it passes through the dies (recall that the volume of the die is considered as a single point) and the ram velocity, will be

² the stable phase of the carbon steel during hot deformation

provided. Based on an input/output model for the plant (velocity vs stroke³ as input, and force vs stroke as output), and using this feedback control framework, a feedback controller will be designed directly to take ram force measurements as feedback signals, and output a velocity profile which most closely follows this feedback relation. After a discussion of microstructural models, the details of the controller design will be presented.

3. Microstructural and Process Models

Plain carbon steels are stable in the austenitic phase at the elevated temperatures characteristic of a hot forging. Typically, the microstructure of austenitic steel is characterized by grain size. Several static and dynamic mechanisms play important roles in the microstructural evolution of the grains before, during, and after, deformation. If the strains in the workpiece are high enough, then the dominant mechanism during deformation is dynamic recrystallization. Models of dynamic recrystallization characterize the microstructural evolution by the recrystallized grain size, d , and the fraction of material recrystallized, χ . A set of simple formula for these quantities is given by Yada (Devadas *et al.*, 1991; Senuma and Yada, 1986) as follows:

$$d = 22,600Z^{-0.27} \quad (1)$$

$$\chi = \begin{cases} 0 & \text{if } \epsilon < \epsilon_c \\ 1 - \exp\left(-.693\left(\frac{\epsilon - \epsilon_c}{\epsilon_s}\right)^2\right) & \text{if } \epsilon \geq \epsilon_c \end{cases} \quad (2)$$

where d has units of microns, and Z is the Zener-Holloman parameter,

$$Z = \dot{\epsilon} \exp\left(\frac{Q}{RT}\right), \quad (3)$$

where Q is an activation energy with the value of 267.1 kJ/mole, and R is the gas constant, with the value 8.31×10^{-3} kJ/mole°K, and $\dot{\epsilon}$ is the strain rate. The temperature T must be

³ Stroke is the displacement of the ram from its initial position.

given in degrees Kelvin. ϵ_c is the critical strain,

$$\epsilon_c = 4.76 \times 10^{-4} \exp\left(\frac{8000}{T}\right) , \quad (4)$$

and $\epsilon_{0.5}$ is the strain increment required for 50% recrystallization,

$$\epsilon_{0.5} = 1.144 \times 10^{-5} d_0^{0.28} \dot{\epsilon}^{0.05} \exp\left(\frac{6420}{T}\right) \quad (5)$$

where d_0 is the initial grain size in microns, (180 μ in this case).

The required strain for 50% recrystallization is typically quite small. Substituting (6) into (5), gives an expression for $\epsilon_{0.5}$ as a function of temperature (initial and final grain sizes are fixed). For the temperature range considered in this study, the $\epsilon_{0.5}$ range is 0.0093 to 0.022. To the resolution of the FEM mesh used in these experiments, that is essentially instantaneous. For the remainder of this study it is assumed that the material recrystallizes completely in the die. Therefore, the microstructure of the workpiece as it exits the die is completely characterized by d . The recrystallized grain size is a function of the strain rate and temperature only. (1) and (3) can be rewritten as a feedback law that gives the strain rate as a function of desired grain size and temperature:

$$\dot{\epsilon} = \left(\frac{d}{22600}\right)^{-3.7} \exp\left(\frac{Q}{RT}\right) . \quad (6)$$

Fig. 1(a) shows the relationship between desired strain rate and material temperature to obtain a given average grain size of 20 μ . As the material cools, the desired strain rate decreases. After the material leaves the die and stops deforming, other mechanisms such as grain growth become extremely important. These effects are neglected here. For the purposes of this study, It is assumed that the workpiece microstructure is essentially frozen as it exits the die. In practice this could be accomplished using rapid quenching. The austenitic phase is unstable at low temperatures. As the workpiece cools, it will undergo phase transformations to ferrite, cementite, and possibly martensite. These complex transformations are highly dependent on the cooling rate. A discussion of this process is beyond the scope of this report. However, the dependence of the microstructure of the transformed phases on the starting

austenitic grain size seems well established. Umemoto (Umemoto, 1990) presents a detailed analysis of these effects. The austenitic grain structure will in any case be "preserved" in the final structure, at least in the sense that a fine-grained structure will become a fine-grained structure.

The macroscopic aspects of the forging are modeled using the commercial forming software, *Antares*, (UES, 1993). This software uses a standard nonlinear implicit iterative finite element based computational strategy to perform forging analysis. The workpiece material is modeled as rigid-viscoplastic, and a shear friction constitutive law is used for characterization of die-workpiece interface behavior.

The example considered in this report is as follows: an austenitic steel billet is extruded through a constant strain rate die. The die is designed following (Srinivasan *et al.*, 1990). Then the relationship between ram velocity and strain rate, and therefore between ram velocity and the temperature of the material in the die, is given by

$$v_{ram} = (5.578 \text{ cm})\dot{\epsilon} = 5.578 \exp\left(26 - \frac{32142}{T}\right) . \quad (7)$$

The resulting feedback law is shown in Fig. 1(b),

The starting grain size is specified to be 180μ . The goal is to reduce the grain size to 20μ throughout the workpiece. If the temperature of the billet is uniform and known, as it would be in an isothermal forging, then the desired feedback control law for the ram velocity is (7). This is the case only if no heat is lost from the billet to the dies, and deformation heating of the billet is neglected. These are not realistic assumptions. Since only the deforming material undergoes dynamic recrystallization, it suffices to consider the temperature of the material in the die. As mentioned earlier, the dimensions of the die are assumed negligible compared to the total length of the workpiece. Therefore, based on this assumption, T is taken as the average temperature of the material in the die. Most of the heat loss by the billet occurs as it sits in the container. The factor that determines how much heat is lost is

the heat transfer coefficient of the lubricating film that coats the billet. This value is a large source of uncertainty for the process. In this study it is the uncertainty in heat transfer coefficient that the feedback controller must compensate for. For simplicity, the value is assumed to be constant over the course of a given extrusion, but to vary between extrusions. Values between 0.0 and 7.3 $\text{kW}/\text{m}^2\text{K}$ were used in the simulations.

Although the average temperature gives a reasonable value to use in the feedback control law, it is not available for measurement. In fact, even extrusion presses used for materials research usually provide little in the way of sensors. One quantity that is accessible is the ram force. Studies have shown that the ram force required to achieve a given ram velocity is correlated with the material temperature (Seetharaman and Chaudhary, 1991). *Antares* provides a model for this relationship, but, as was previously mentioned, *Antares* is too slow for on-line computation.

In the sections which follow, the use of *Antares*' behavior to train the ANN to emulate the ideal control law for the forging process will be discussed.

4. A Brief Introduction to Neural Networks in Control

Over the past few years, there has been a flurry of activity in the area of applying neural networks to identification and control of nonlinear systems, ((Ahmed, 1994; Chen and Khalil, 1991; Chu and Shoureshi, 1991; Hunt *et al.*, 1992; Lin, 1993; McGrane *et al.*, 1994; Narendra and Parthasarathy, 1990; Teixeira *et al.*, 1991; Hoskins *et al.*, 1992; Davis *et al.*, 1993; Narendra *et al.*, 1995), for example).

A major premise in much of the literature pertaining to the application of neural networks to the adaptive control of nonlinear systems is that the nonlinear systems models are feedback linearizable with stable zero dynamics, ((Chen and Khalil, 1991; Lin, 1993; Narendra and Parthasarathy, 1990; Narendra *et al.*, 1995)). The reason for this assumption is that a simple,

causal, stable controller may be solved for in this case. This assumption is unnecessary, provided that the neural network model has the property that its targeted outputs define a reachable trajectory from allowable (within constraints) control trajectories. Feedback linearizability of a nonlinear model is sufficient, but not necessary for the model in question to be to satisfy this form of reachability requirement.

For nonlinear control systems models, the concept that the output trajectories are reachable from some input trajectory, or, equivalently, that target tracking can be achieved for some permissible input, is strongly related to the invertibility of the model. Invertibility of an ANN is the ability of the network to trace backwards through itself, starting from an arbitrary output, to reach an input which generated that output through the network.

Although the method of iterative inversion (Linden and Kindermann, 1989) has been applied in several applications, (Hoskins *et al.*, 1992; Davis *et al.*, 1993), for neural network models, it is not necessary to first train a forward ANN model, and then invert it, in order to implement control, since inverse models may be trained directly from the observed input and output data. This is the approach followed in this work.

In contrast to works such as (Lin, 1993) and (Ahmed, 1994), in this work, aside from the microstructural models, there are no apriori assumptions made about the internal model structure of the plant. Direct neural network control design will be implemented without requiring the identification of a forward plant model, or a parameterized control law, as required in (Lin, 1993; Ahmed, 1994; Narendra and Parthasarathy, 1990; Hoskins *et al.*, 1992; Davis *et al.*, 1993), for example.

There are several important decisions a modeler makes in choosing a configuration for a neural network model. The number of hidden layers, the nonlinear activation function acting at each layer, whether or not the model is recurrent, the learning algorithm, the amount of training data, the learning rate, and the error tolerance must all be chosen. These choices

will influence the speed and accuracy of the model, as well as its ability to generalize.

Teixeira et. al., (Teixeira *et al.*, 1991), derive a method, in terms of the size of the training data and the number of inputs and outputs, for calculating the minimum number of hidden elements in a neural network, with either one or two hidden layers, in order for the neural network to represent an input/output relation with no error.

There are some rules of thumb for choosing some of these other parameters, but sometimes it is necessary to try several configurations before a good neural network model is derived.

The next section discusses the use of neural networks in a feedback configuration for real-time control of a forging process.

5. Feedback Control in Forging Using Neural Networks

Fig. 2 depicts a schematic for an extrusion press in use at Wright Laboratory. It is of note that there are two force measurement devices: the hydraulic fluid pressure at the ram, and the load cell at the nose. Either of these force measurements may be used (suitably translated) to calculate the ram force. The plan is to implement the trained ANN controller in feedback with the forging process as in Fig. 3(a).

For the simulation described in this report, the trained ANN was put in feedback with *ANTARES*, to see how well the ANN could predict velocities based on the *temperature* feedback law, (7), (as determined by the FEM program). The next subsection describes the choice of network architecture.

5.1. *Choice of Network Architecture*

After some reasoning, (as well as some trial and error), the choice of network architecture for the simulation described in this report is: two inputs (force, stroke), one output, (velocity),

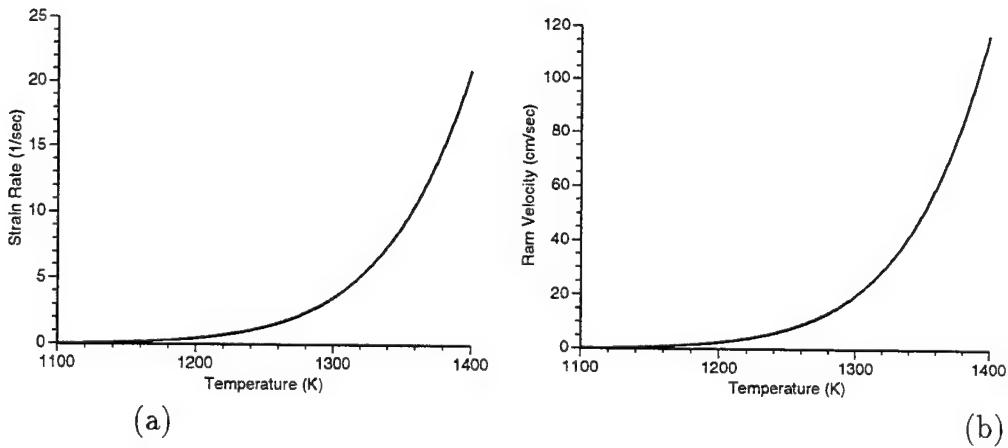


Fig. 1. (a) Strain Rate Vs. Temperature; (b) Velocity Vs. Strain Rate

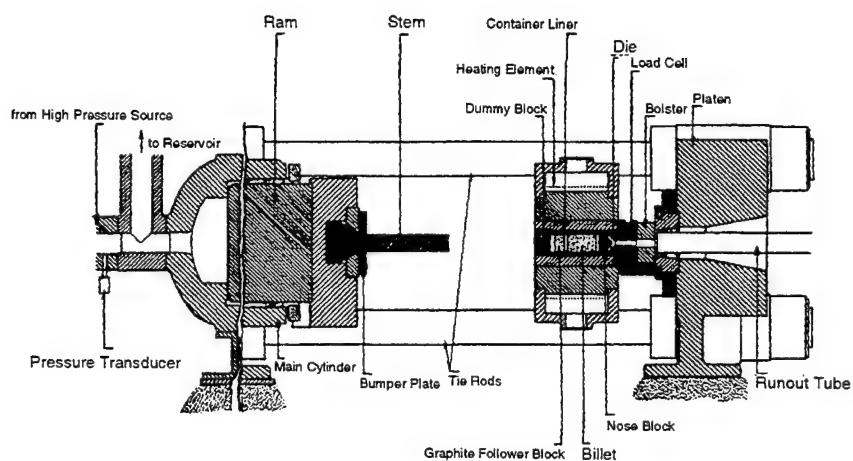


Fig. 2. Extrusion Press

a linear activation function at the output, two hidden layers of 16 nodes each, and hyperbolic tangent nonlinearities for the activation functions to the two hidden layers.

The ANN controller was trained on five independent data sets (representing heat transfer coefficients between the billet and the die/container of 0.0, 0.29, 1.5, 2.9, and $7.3 \text{ kW/m}^2\text{K}$) which contained *Antares*-generated velocity, stroke, and force profiles, (each respective profile containing 32 values). The initial billet temperature was set to 1255°K , and the initial die and container temperature was set to 311°K . The ANN was trained using *Matlab*, for 10,000 epochs, using back propagation with momentum and an adaptive learning rate.

5.2. ANN Training Results

The configuration used to train the ANN is shown in Fig. 3(b). The training is done directly on the FEM plant model, and does not require a separate ANN representing the plant. The raw training data is shown in Fig. 4(a).

Notice the trend in the force data for the various data sets relative to the first data set. For the purposes of training, this trend was removed from the data and all the data was normalized. The removal of this trend greatly improved the training, and so was incorporated into the controller design for consistency in implementation. The stroke data was also scaled by a factor of 1/10 to reduce its influence on the controller. These modifications were applied to the input data actually used to train the ANN. A plot is shown in Fig. 4(b).

The result of the ANN training is shown in Fig. 5. The neural network controller performs quite well in tracking the training data generated by *Antares*.

5.3. ANN Feedback Control Simulation Results

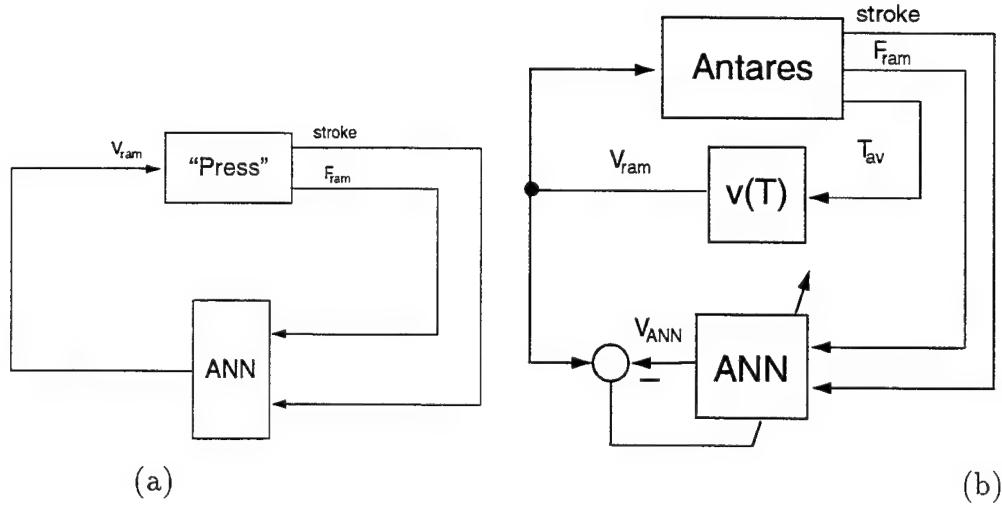


Fig. 3. (a) ANN Feedback Control of Forging; (b) ANN Controller Training Configuration

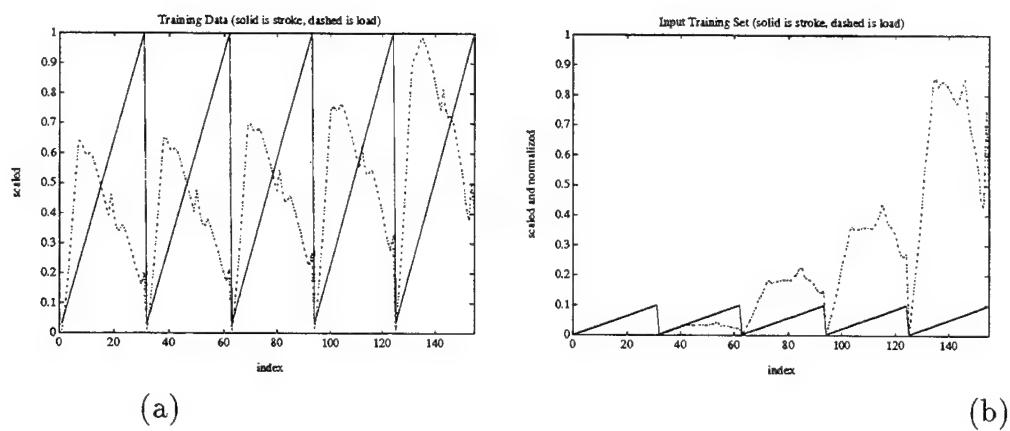


Fig. 4. (a) Training Data; (b) Modified Training Data

To validate the network, Fig. 6(a) represents a comparison of a *desired*⁴ *Antares* velocity profile, (based on data which was not part of the training data, due to a modified heat transfer coefficient), to the velocity profile output by the ANN controller based on the same force profiles *Antares* generated. This is an *open loop* validation of the ANN controller, and the results are quite good.

In Fig. 6(b), the ANN is used in closed loop with *Antares*, in that here, the velocity profile seen by *Antares* is that generated by the output of the ANN controller, and a comparison is made between the velocity output of the ANN in closed loop with the FEM simulation, and the *desired* velocity profiles predicted step by step by *Antares*, (based on the previous ANN generated velocity). Again, the closed loop results are quite good.

Just to test the sensitivity of these results to initial velocity, the simulation was run using initial velocities which were three times and one third the actual initial value for that data set, and the neural network was still able to perform well in tracking the optimal velocity profile. The sensitivity results are shown in Fig. 7.

6. Discussion and Conclusions

In this work, a simulation of direct neural network control of a round to round extrusion of plain carbon steel has been presented. The simulation results are quite promising.

This work is important, not only for its potential application to an important industrial problem, but because of the novel use of the neural network in direct controller design of a complex system for which no real-time computable analytical model exists.

In the example presented, force measurements were used as a measurement signal to drive the ANN controller. The main reason for using force measurements was their availability.

⁴ the desired velocity profile is given by the feedback law determined by the microstructural models, (7), using the average temperature of the material passing through the die, as generated by *Antares*

The temperatures which are necessary to directly drive the velocity control, (7), are not measurable. As sensor technology improves, or as presses become more heavily instrumented, it is expected that the method described in this work will generalize.

7. REFERENCES

Ahmed, M.S. (1994). Block partial derivative and its application to neural network based direct model reference adaptive control. *IEE Proceedings, Part D: Control Theory and Applications* **141**, 305–314.

Berg, J.M., A. Chaudhary and J.C. Malas (1995a). Optimal open-loop control of a hot forming process. In: *Proceedings of NUMIFORM '95*. pp. 539–544.

Berg, J.M., R.J. Adams, J.C. Malas and S.S. Banda (1994). Design of ram velocity profiles for isothermal forging via nonlinear optimization. In: *Proceedings, American Control Conference*. pp. 323–327.

Berg, J.M., R.J. Adams, J.C. Malas and S.S. Banda (1995b). Nonlinear optimization-based design of ram velocity profiles for isothermal forging. *IEEE Trans. Control Applications* pp. 269–278.

Chen, F-C. and H.K. Khalil (1991). Adaptive control of nonlinear systems using neural networks—a dead-zone approach. In: *Proceedings, American Control Conference*. pp. 667–672.

Cheng, H., R.V. Grandhi and J.C. Malas (1994). Design of optimal process parameters for non-isothermal forging. *International Journal for Numerical Methods in Engineering* **37**, 155–177.

Chu, S.R. and R. Shoureshi (1991). A neural networks approach for identification of continuous-time nonlinear dynamic systems. In: *Proceedings, American Control Conference*. pp. 1–5.

Davis, D.T., Z. Chen, L. Tsang, J-N Hwang and A.T.C. Chang (1993). Retrieval of snow parameters by iterative inversion of a neural network. *IEEE Trans. Geoscience and Remote Sensing* **31**(4), 843–852.

Devadas, C., I.V. Samarasekera and E.B. Hawbolt (1991). The thermal and metallurgical state of steel strip during hot rolling: Part iii, microstructural evolution. *Metallurgical Transactions A* **22A**, 335–348.

Hoskins, D.A., J-N Hwang and J. Vagners (1992). Iterative inversion of neural networks and its application to adaptive control. *IEEE Trans. Neural Networks* **3**(2), 292–301.

Hunt, K.J., D. Sbarbaro, R. Zbikowski and P.J. Gawthrop (1992). Neural networks for control systems—a survey. *Automatica* **28**(6), 1083–1112.

Lin, J. (1993). Direct adaptive output tracking control using multilayered neural networks. *IEE Proceedings, Part D: Control Theory and Applications* **140**, 393–398.

Linden, A. and J. Kindermann (1989). Inversion of multilayer nets. In: *Proceedings, International Joint Conference on Neural Networks*. Vol. 2. pp. 425–430.

Malas, J.C., R.D. Irwin and R.V. Grandhi (1993). An innovative strategy for open loop control of hot deformation process. *Journal of Materials* **2**(5), 703–714.

McGrane, D., R. Smith and M. Mears (1994). A study of neural networks for flight control. In: *Proceedings, American Control Conference*.

Malas, J.C., R.D. Irwin and R.V. Grandhi (1993). An innovative strategy for open loop control of hot deformation process. *Journal of Materials* **2**(5), 703–714.

McGrane, D., R. Smith and M. Mears (1994). A study of neural networks for flight control. In: *Proceedings, American Control Conference*.

Meyer, D. and H.N.G. Wadley (1993). Model-based feedback control of deformation processing with microstructure goals. *Metallurgical Transactions B* **24B**, 289–300.

Narendra, K.S. and K. Parthasarathy (1990). Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Networks* **1**(1), 4–26.

Narendra, K.S., J. Balakrishnan and M.K. Ciliz (1995). Adaptation and learning using multiple models, switching, and tuning. *IEEE Control Systems Magazine* pp. 37–51.

Seetharaman, V. and A. Chaudhary (1991). Prediction of 3-d material flow in round to rectangular extrusion. Technical Report WL-TR-91-4018. USAF Technical Report. prepared by A. Chaudhary and S. Doraivelu.

Senuma, T. and H. Yada (1986). Microstructural evolution of plain carbon steels in multiple hot working. In: *Proceedings of the 7th Riso International Symposium on Metallurgy and Materials Science* (S.S. Hansen et al., Ed.). Riso National Laboratory, Roskilde, Denmark. pp. 547–552.

Srinivasan, R., J.S. Gunasekera, H.L. Gegel and S. Doraivelu (1990). Extrusion through controlled strain rate dies. *J. Mater. Shaping Technol.* **8**, 133–141.

Teixeira, E., K. Loparo and F.A.C. Gomide (1991). Design of multi-layer neural networks for accurate identification of nonlinear mappings. In: *Proceedings, American Control Conference*. pp. 14–15.

Tibbetts, B. and J. Wen (1995). Application of modern control and modeling techniques to extrusion processes. In: *Proceedings of the 4th IEEE Conference on Control Applications*. Albany, NY. to appear.

UES, Inc. (1993). *Antares User's Manual Version 3.0*. United Engineering Systems. Dayton, OH.

Umemoto, M. (1990). Mathematical modeling of phase transformation from work-hardened austenite. In: *Proceedings of the International Symposium on Mathematical Modelling of Hot Rolling of Steel* (S. Yue, Ed.). Hamilton, Ontario, Canada.

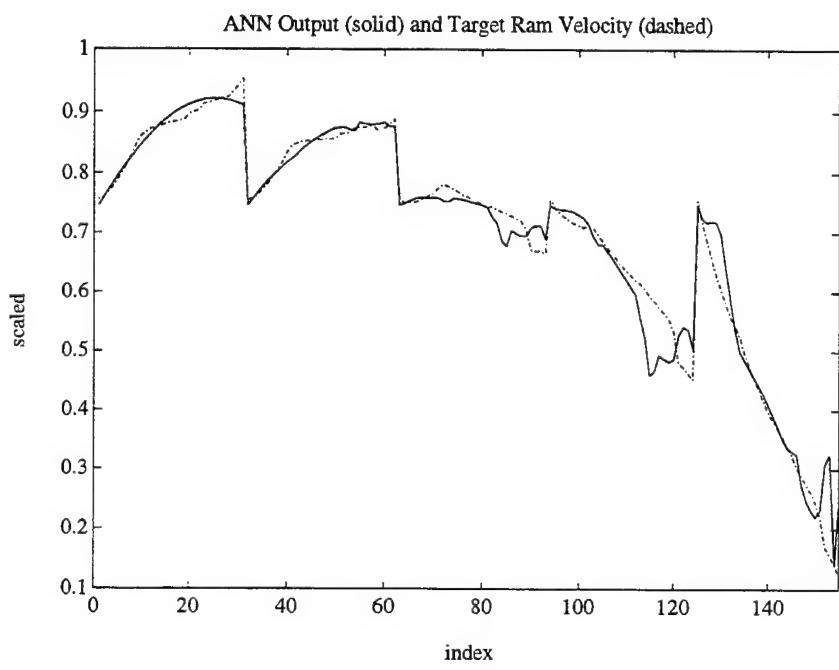
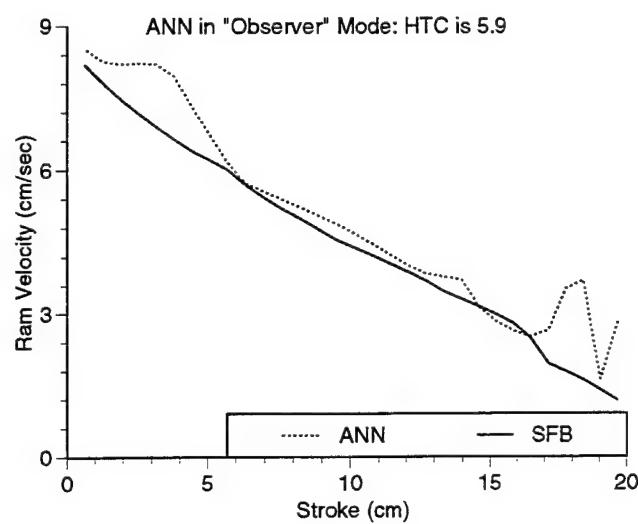
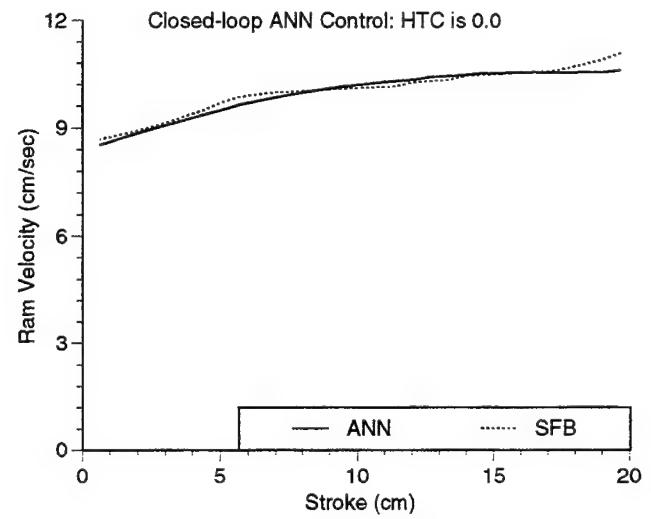
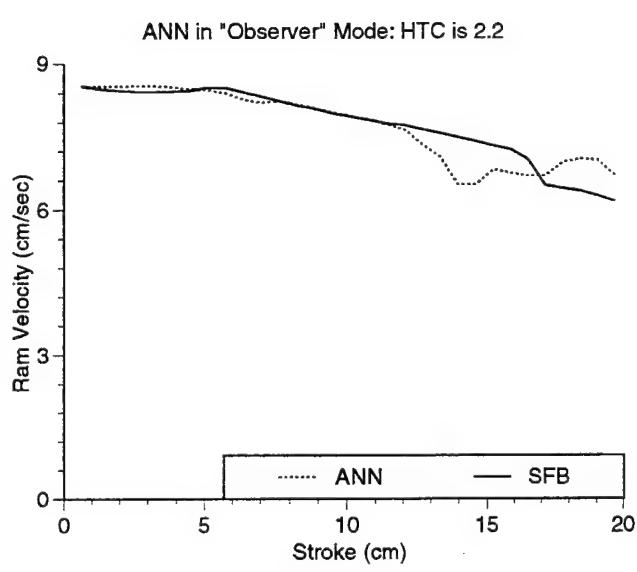
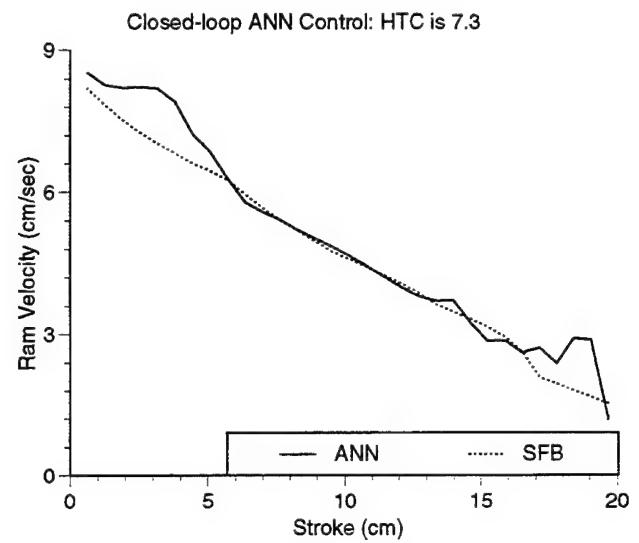


Fig. 5. ANN Training Results



(a)



(b)

Fig. 6. (a) Open Loop Validation of ANN; (b) Closed Loop Control of Antares

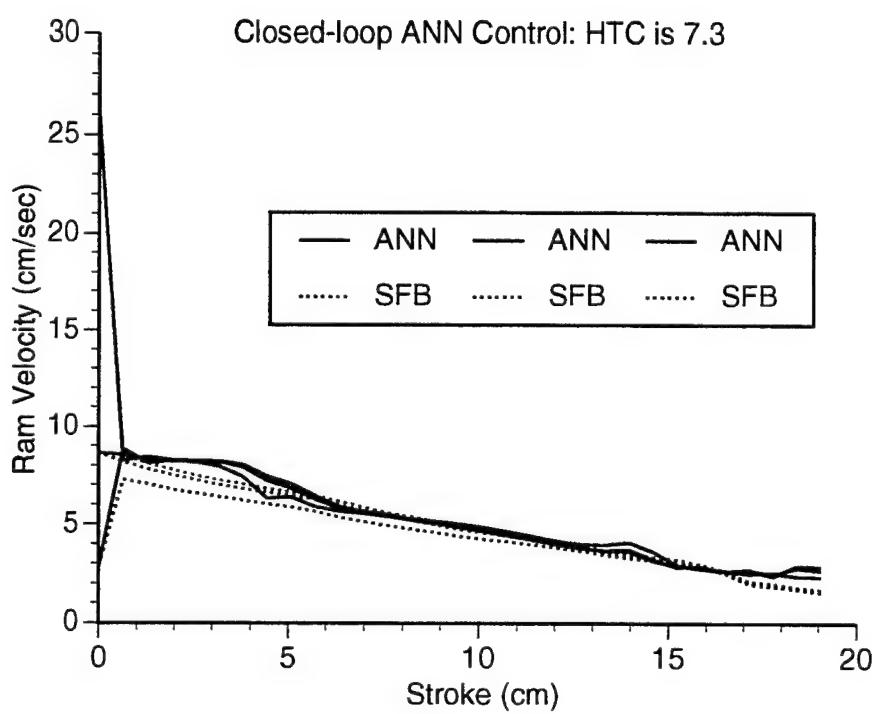


Fig. 7. Initial Velocity Sensitivity

Documentation of Separating and Separated Boundary Layers

Terrence W. Simon
Professor
Department of Mechanical Engineering

and

Ralph J. Volino
Research Assistant
Department of Mechanical Engineering

University of Minnesota
Minneapolis, MN 55455

Final Report for:
Summer Research Extension Program
Wright Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, Washington, D.C.

and

University of Minnesota

December 1995

DOCUMENTATION OF
SEPARATING AND SEPARATED BOUNDARY LAYERS

Terrence W. Simon
Professor
Ralph J. Volino⁺
Research Assistant
Department of Mechanical Engineering
University of Minnesota

Abstract

The low-pressure turbine of an aircraft engine operates with a low chord Reynolds number. As such, there are regions of strong streamwise acceleration and diffusion effects. This results in extended regions of transition from laminar to turbulent flow and large zones of flow separation. In response to a need to learn more about the mechanisms that lead to transition and separation in the engine environment, a low-Reynolds-number-flow study was initiated during the Summer of 1994 at Wright Labs. In this project, a low-pressure turbine airfoil cascade was installed in a wind tunnel. To simulate the engine environment, high background turbulence was imposed on the flow and a device for imposing passing wakes upon the flow was fabricated. A program for measurement of the characteristics of the boundary layer; laminar-like or turbulent, separated or attached, was initiated. The Summer project resulted in an effective start on the problem but considerably more remained to be done. This report documents the subsequent work on this project. At the University of Minnesota, an easily accessible facility which had the essential elements of the low-pressure turbine flow was designed, built, and qualified. This facility now provides a convenient means for documenting the flow and developing measurement techniques. The Wright Lab experimental program continued with the completion of the construction, the implementation of the turbulence generating device, and the qualification of the tunnel. Both facilities are now producing data to the program. These data are summarized herein. The University of Minnesota facility has generated pressure profiles for various cases of different Reynolds number and turbulence intensity. The Wright Labs facility has give pressure profiles for various Reynolds numbers. The two facilities are now ready for detailed measurements within the boundary layers.

⁺ Presently an Assistant Professor at the U.S. Naval Academy

DOCUMENTATION OF SEPARATING AND SEPARATED BOUNDARY LAYERS

Terrence W. Simon
Ralph J. Volino

Introduction

Compressor and turbine design models have been moderately successful in predicting losses and heat transfer rates in the high-pressure components of the gas turbine where chord Reynolds numbers, Re_c , are large and separation and transition regions are small. When applied to low chord Reynolds number flows, where long transitional zones and large separation bubbles are present, these models fail. Being able to predict transition and separation under low-Reynolds number operation has recently taken on increased importance for improved designs of gas turbine engines. To illustrate, in the low pressure turbine where the airfoils are aft-loaded, 90% of the blade suction surface can be covered with transitional boundary layer flow (Mayle, 1991). Also, when the chord Reynolds number on an airfoil is decreased, the boundary layer becomes more likely to separate. Separation is expected for Re_c values under 400,000 (Sharma, Ni, and Tanrikut, 1994), although, clearly, there is more to the prediction of separation than the knowledge of Re_c . For instance, elevated free-stream turbulence would reduce the separation Reynolds number. The dependence on turbulence level and scale is presently unknown. The effects of disturbances due to wakes generated by upstream airfoil rows on this separation Reynolds number are even more uncertain. Wakes may result in an instantaneous increase in turbulence level. Also, if the flow were behaving in a quasi-static manner, one would expect wakes to change, momentarily, the angle of attack to an off-design angle, favoring increased separation on the suction surface. Because wakes pass at a high frequency, the quasi-static assumption may be unwarranted. Needed are transient measurements of the boundary layer flow over the downstream portions of the suction surface, ensemble-averaged on the wake passage event. Such measurements provide information about the state of the boundary layer and the receptivity of the boundary layer to external disturbances, including those from the passing wakes. Possible wake generators may be cylinders, representing wake turbulence with large-scale turbulence; thin plates oriented parallel to the flow, representing wakes which consist of decaying boundary layer turbulence; or actual airfoils. Comparisons of measurements in cascades and rotating rigs indicate that losses and heat loads are higher in the unsteady flow than in steady flow (Sharma et al., 1994, Hodson, 1983, Blair et al., 1988, Doorley et al., and Sharma et al. 1990).. Sharma et al.,

1994 proposed making such comparisons using scaling on a relative time scale to capture this effect. The appropriate ratio is the wake passage period divided by the transit period for fluid flow through the airfoil row. Cases of similar values of this ratio display similar augmentation due to unsteadiness. Such scaling is applied in the Wright Lab experiment. Operation at low Reynolds numbers would create separation zones, allowing a study of the effect of the wakes on the incipience to the separation process, on the free-shear layer transition length, and on the separation bubble length. Under low-Reynolds-number conditions, upstream wakes can result in smaller separation zones and lower losses relative to steady flow. Presently, no model has been developed to capture this effect (Sharma et al., 1994). Thus, wake effects on the separated flow must be included for a low-Reynolds number study to be accurate and complete. This may be the first experiment with a detailed investigation of the low-pressure turbine flow boundary layer including wake effects. It is important that these tests are done well. Hot-wire anemometry measurements can be made in the free-shear layers over the separation bubbles without affecting the flow. Measurements within the separated flow zone are difficult, but possible, and needed. With these measurements, one could assess whether the change in incidence angle during the wake passing event is important or whether the main wake-related effect is that of turbulence washing over the boundary layer and free-shear-layer flows. Thus, careful documentation of the turbulence and wake disturbance effects is needed in support of design model development.

Objectives

One objective of this program is to coordinate experiments at the University of Minnesota and Wright Labs. to optimize this important test program.

Objectives of the Wright Laboratory Experiment

The objectives of the Wright Lab. program are to determine the transition and separation zone locations for a particular low-pressure airfoil under representative conditions, including an assessment of the effects of free-stream turbulence and of wake passings on the locations of these zones. A secondary objective is to document the pre-separation boundary layer receptivity to external disturbances and the instability of the free-shear layer over the separation zone. In doing so, several representative cases are to be investigated, the transition and separation zones for each are to be located, and measurements are to be taken in the pre-separation boundary layer and in the separation zone. For cases with wakes, this documentation is to be given for various times within the wake passing period.

Objectives of the University of Minnesota Experiment

The University of Minnesota experimental program is configured to support the cascade measurements, but in a simple, single-airfoil test section where probe access is easy and the geometry is more simple. Though a cascade is not used, the flow will simulate most features of the flow in the Wright Lab. cascade, for the curvature and pressure gradient will be replicated. Additionally, the free-stream turbulence will be simulated and the wake passage events will be added in the near future. In this simulated flow, detailed hot-wire measurements will be made to develop a means by which the waveform of the hot-wire time-trace will be used, in conjunction with flow visualization and static pressure profile measurements, to determine the state of the flow. Some turbulent transport terms will be measured in the near-wall region to verify that the boundary layer flow is truly turbulent (contains turbulent production, not just the unsteadiness of the external turbulence in the free-stream), or is free of significant turbulence production. This, we have done over the last six years in similar, attached flows but with constant-pressure boundary layers or with favorable pressure gradients. Extensions of our past techniques will be used in flows with the imposition of an adverse pressure gradient and in the investigations of incipient-to-separation and separated flows. Results from this work will be used by the University of Minnesota and the Wright Lab. researchers in the Wright Lab. cascade facility. Additionally, computational efforts at the Wright Lab. will be made to compute this flow to test present schemes for the prediction of transition and separation on low-pressure turbine airfoil surfaces. The University of Minnesota experiments, the Wright Lab. experiments, and the Wright Lab. computations constitute the overall program. The experiments will provide comparison data for the computations and turbulence data for improvements of the turbulence transport predictions within the computational models. Researchers of the team will be looking for techniques for improving the turbine performance which may come about by reduction of boundary layer separation zones. Thus, results may include improved future turbine designs and improved design models. If the eventual results of this program can lead to even a percent improvement in the engine performance, as they have a strong likelihood of doing, the return on investment for this work is enormous. There is considerable opportunity for payback when working on the low-pressure turbine. It has traditionally not received the level of attention given to compressor or high-pressure turbine blading. The low-pressure turbine has recently been targeted as a section of the engine in need of research attention (Sharma, Ni, and Tanrikut, 1994).

Facilities

The Wright Lab Cascade Facility

The Wright Lab. experiments were conducted in the AFIT cascade facility. This facility has been documented in numerous papers and theses from the Lab. Recent modifications to the facility are documented in the report of Summer 1994 research by Simon and Volino. This wind tunnel is driven with a centrifugal blower operating in the suction mode. Flow velocity control is with a motor controller, with inlet dampers on the fan, and with a bypass vent on the ductwork between the test section and the fan. A plan view of the wind tunnel is given as Fig. 1. The cascade test section consists of four geometrically identical blades of chord length 11.4 cm (4.5 inches).

Turbulence Generator

Upstream of the cascade is a turbulence generation device which consists of a passive square grid of 13 mm by 13 mm (0.5 inch by 0.5 inch) square bars arranged with a 25.4 mm (1.0 inch) center-to-center spacing. This resides 1.5 m (57 inches) upstream of the cascade row. A distance of 0.6 m (24 inches) downstream of the passive grid is an active jet grid with 6 tubes each with 6 blowing holes distributed and oriented as shown in Fig. 2. The passive grid/jet grid arrangement is a replica of one described by Sahm and Moffat (1992). This type of generator was reproduced and installed in the University of Minnesota facility.

Wake Generator

At a distance of 7.6 cm upstream of the leading edge of the cascade row is a series of 6 cylinders which are traversed across the tunnel cross-section in the transverse direction. These cylinders simulate the wakes that are generated by the airfoil row which resides just upstream of the airfoil row of interest in an actual turbine. The cylinders are 9.5 mm (0.375 inch) in diameter and are separated by the same transverse spacing as that of the airfoils, 91.7 mm (3.611 inches). These cylinders are driven by a device which is capable of translating them at selected velocities from 0.5 m/sec to as high as 5.5 m/sec. The total translation distance is 43 cm (17 inches). Attached to the drive mechanism is a photo-diode sensor which indicates when the translation runner has moved a distance of 10 cm (4 inches). The signal from this sensor is used to activate the data acquisition trigger. The duration of the measurement portion of the translation is from one to five wake passings. A similar wake generator is being fabricated for installation in the University of Minnesota facility. Wake passing data should be taken at Minnesota during the Summer of 1996.

The measurement techniques developed at the University of Minnesota have been and will continue to be implemented in the Wright Lab. facility and program. Such implementation will be done by communication with the Wright Lab. group and by the present researchers during visits to Wright Lab.

The University of Minnesota Facility

The University of Minnesota facility is complementary to the cascade facility at Wright Lab. The flow passes through the same turbulence generators as at Wright but then passes through a channel between two curved walls, Fig. 3. The leading edges of the walls are configured to simulate the leading edge/stagnation-line regions of two adjacent blades. The two (suction and pressure) downstream walls are bent and oriented so that the curvature and pressure gradient profiles of the cascade are replicated. The facility has an airfoil aspect ratio of 6, thus, the effects of the endwall flow on the mid span boundary layers are minimal. Also, the simpler geometry of the facility affords easier access for flow visualization and detailed measurements.

About the measurements:

The first measurements, already taken and presented herein, are profiles of surface static pressure. Next, measurements will be taken very near the surface to determine zones of flow reversal. A precursor to this was a program of near-wall measurements to assess the utility of such measurements for the direct computation of wall shear stress and heat transfer coefficient. This is documented in Qiu et al. (1995). More remains to be developed here, however. This development will continue during the Spring of 1996. Because the hot-wire does not discriminate flow direction for a single sample, the entire flow waveform must be recorded. The waveform will have sufficient temporal resolution to show flow reversal among other important events. After the near-wall measurements are taken, the evolution of the attached boundary layer will be documented, including documentation of such turbulence quantities as elements of the Reynolds shear stress tensor. Following this will be a similar study, but applied to the free shear layer over a separated zone. Throughout, flow visualization using smoke injection, oil and lampblack, the ink-dot technique, or the smoke-wire technique, all familiar to our lab, will be used to complement the measurements.

Hot Wire Anemometer

Instantaneous local velocities are measured using a TSI Model 100 (IFA-100) Intelligent Flow Analyzer, constant-temperature anemometer with a TSI Model No. 1210-T1.5 Tungsten single-wire, hot-wire probe, a TSI Model No. 1243-20 platinum cross-sensor, hot-film probe, or a

TSI Model No. 1299BM triple-sensor probe. The cross-sensor probe is used to measure turbulent shear stress. It has two $51 \mu\text{m}$ (2 mil) diameter sensors, mounted perpendicular to one another, parallel to the cascade endwall, and 45° to the airfoil surfaces. The single-wire sensor is used for indicating the state of the boundary layer on the suction surface of the airfoil by traversing it very near the surface, measuring the fluctuation intensity and interrogating the waveform to detect signs of flow reversal. The single-wire probe has one $4 \mu\text{m}$ (150 μinch) diameter wire, mounted perpendicular to the flow and parallel to the convex wall. Each hot-wire output signal, an analog voltage, is amplified and filtered using a TSI Model No. 158 signal conditioner. Each is read with a data acquisition unit. The triple-sensor probe is used to obtain 3-D measurements. It has a 90° bend with six straight prongs (see Fig. 4). Three films sensors of $51 \mu\text{m}$ (2 mil) diameter are mounted perpendicular to one another and with a 35° inclination angle.

Surface Static Pressure

The surface static pressures are read using static pressure ports installed on a wall surface. They are connected to the diaphragm of a Validyne, 860 Pa- (3.5 inch)- maximum-pressure-difference, variable-reluctance, pressure transducer driven by a Validyne model CD-15 Carrier Demodulator. The pressure transducer provides an analog voltage signal which is read with the data acquisition unit.

Temperatures

A single Type E thermocouple is used to record the flow temperatures.

Data Acquisition

All the data acquisition is with a Norland Prowler (now High-Techniques) high-Speed Data Acquisition System. This unit has a 12-bit A/D converter which allows simultaneous sampling of two channels per unit (6 channels in the Lab.). Though the bulk of the Minnesota work will be with the single wire, X-wire measurements may be included to help determine the state, laminar or turbulent, of the boundary layer. Sampling rates of as high as 100,000 samples/sec can be achieved and record lengths of up to 4096 readings per channel can be acquired per 2-channel unit. Data acquisition is controlled with a PC using software written in C. Equipment at the Wright Labs. is similar. It is described in detail by Simon and Volino (1994).

Power spectral Density (PSD) measurements may also be taken in regions where the flow is not reversing. Such data are acquired in three sections. The first section is acquired with a 100 kHz sampling rate and low-pass filtered at 10 kHz. The second section is acquired with a 10 kHz sampling rate and low-pass filtered at 1 kHz. The third section is acquired at 1 kHz with low-pass filtering at 100 Hz. For each section, 20 traces of 4096 points are digitized. Acquiring the spectra in sections allows better resolution of both high and low frequencies, maximizing the quality of the spectrum, given the limited amount of data that can be acquired and stored in the digitizer for each record. Instrumentation for PSD measurements is similar whether at the University or the Wright Lab. Spectral measurements have proven useful in past work for differentiating between badly-disturbed, non-turbulent flow and flows with turbulence production at the wall.

The approach flow turbulence levels and PSD for three components of velocity, streamwise, cross-stream, and spanwise, have been taken using a TSI Model 1299BM triple-sensor, hot-film probe. An example of these data is shown in Fig. 5. Integral length scales computed from these three spectra are 2.6, 2.1, and 2.2 cm for u, v, and w velocity components, respectively.

Status of the Program

At Wright Labs.

The entire test facility has been modified to include the new airfoils and the new turbulence generator. The new wake generator has been fabricated but the test program is not sufficiently far along for the generator to be installed in the tunnel. The new airfoils have been instrumented with static pressure taps. A sheet with surface-mounted thin-film sensors has been fabricated and is ready for implementation when the program is ready for it. The thin-film sensor sheet has 30 hot-film elements attached to bus wires. The wake generator slider mechanism, the device which drives a row of cylinders or airfoils ahead of the test section, has been designed, constructed, and partially qualified. A photo-diode device to create a trigger signal for the data acquisition has been constructed and checked out. All data acquisition methodology and software has been formulated, transferred to Wright Labs. personnel, and implemented. The tunnel modification is complete. The tunnel has been adjusted to minimize non-uniformity of flow from one passage to the next and non-uniformity of the approach flow. The approach flow characteristics are now suitable. The aspect ratio of the blades is 1.0 which results in an influence of the endwalls on the flow. It seems that this influence may even extend to the centerspan in some cases, thus, careful attention to this effect is needed. Both of the

present investigators have visited the Wright Labs. twice during the period of this grant to take part in the qualification of this tunnel and the senior investigator visited a third time for continued qualification. Qualification visits first entailed the assembly of the facility and acquisition of preliminary check-out data along with the transfer to Wright personnel of instruction about the installed software. Later visits involved measurements and rig modification to achieve improved uniformity and attempts at dealing with the small aspect ratio effects by manipulation of the flow with boundary layer fences. At the Wright Labs., Dr. Richard Rivir and Chris Murawski have continued with the development and qualification. Aside from the aspect ratio concern, the facility is now ready to go and pressure profile data have been taken.

At the University of Minnesota

The University of Minnesota work first involved the reconstruction of a wind tunnel for use with the low-pressure turbine. This was done in conjunction with a low-Reynolds-number program under NASA funding. The tunnel is now complete with the installation of a turbulence generation system and a representative low-pressure turbine airfoil shape. A wake generator fashioned after the Wright Lab. design is presently being fabricated and will allow introducing wakes to the flow. It will be ready for the test program late this Spring. Thus far, the tunnel has allowed a study of the effects of Reynolds number and turbulence intensity on the pressure profile. Careful documentation of the flow with spectral measurements of the three components of the approach flow turbulence is underway. Such measurements include documentation of the decay of turbulence, as needed by computational persons. It is hoped that not only will the data on flow stability from these two rigs be informative in their own right, but the measurements will represent a complete and accurate data set for judging analytical and numerical design methodology. Presently, the approach flow documentation is being completed and a program for measurements within the boundary layer is being charted.

Results to date:

Pressure profiles have been documented for the cases shown in Table 1. These are represented by the cases shown in Figs. 6 and 7. Chord Reynolds numbers presented herein are based upon axial chord length and inlet velocity. For reference, the three Reynolds numbers in Fig. 6 would be 54,000, 98,000 and 196,000 if based on exit velocity and suction surface length. Figure 7 shows that a low Reynolds number case can be dramatically influenced by the level of free-stream turbulence intensity. The highest TI case (10%) appears to be free of boundary layer separation except for a small possible bubble at 80% of chord length. As the turbulence level is

reduced to 3% the separation zone grows to include, it appears, the length $75\% < x/c < 85\%$. Finally, at the lowest value of turbulence, $<1\%$, the separation zone has grown to last 45% of the chord length. Of course, these statements are speculative at this point for we wait for detailed measurements within the flow to confirm that the variations of static pressure represent separation and not some other effect such as a change in the process by which the boundary layer is passing through transition. In Fig. 6 three cases of different Reynolds number are compared for one turbulence level, 3%. Here one can see that as the Reynolds number drops to 40,000, a small separation bubble emerges, and, as it drops further to 22,000, this bubble enlarges considerably, apparently consuming the range $60\% < x/c < 90\%$. Again, more will be learned about these traces when detailed boundary layer measurements are taken.

Summary

The sponsored extension program has allowed opportunities to travel to Dayton for assembly, experimentation, and transfer of information and has allow additional development at the University of Minnesota. The turbulence generation and wake generation development at Wright Labs. has been implemented into the Minnesota program and the instrumentation developed at Minnesota has been implemented into the Wright Lab. program. The communication will continue via e-mail, fax, and phone, now that the close link has been established. In a recent review meeting on the low-pressure-turbine efficiency problem at the NASA Lewis Research Center, the two programs were reviewed before a group of researchers and industry representatives. During the proceedings, it became clear that the two projects discussed herein are highly complementary and that both are major cornerstones to the concerted efforts in the U.S. today on the problem.

References

Blair, M. F., Dring, R. P., and Joslyn, H. D., 1988, "The Effects of Turbulence and Stator-Rotor Interactions on Turbine Heat Transfer, Part I, Design Operating Conditions," ASME Paper #88-GT-125.

Doorley, D. J., Oldfield, M. L. G., and Scrivener, C. T. J., "Wake Passing in a Turbine Rotor Cascade," AGARD CP-390, Paper No. 7, Bergen, Norway.

Hodson, H. P., 1983, "The Development of Unsteady Boundary Layers in the rotor of an Axial-Flow Turbine," AGARD Proceedings No. 351, *Viscous Effects in Turbomachines*.

Mayle, R. E., 1991, "The Role of Laminar-Turbulent Transition in Gas Turbine Engines, ASME J. Turbomachinery, Vol. 113, pp. 509-537.

Qiu, S., Simon, T. W., and Volino, R. J., 1995, "Evaluation of Local Wall Temperature, Heat Flux, and Convective Heat Transfer Coefficient from the Near-Wall Temperature Profile, *Turbulent Heat Transfer*, ASME Vol. 285, N. H. Anand, Ed., presented at the IMECE Conference, San Francisco, CA.

Sahm, M. K., and Moffat, R. J., 1994, "Turbulent Boundary Layers with High Turbulence: Experimental Heat Transfer and Structure on Flat and Convex Walls," Report No. HMT-45, Thermosciences Division, Mechanical Engineering Department, Stanford University, 1992

Sharma, O. P., Pickett, G. F., and Ni, R. H., 1990, "Assessment of Unsteady Flows in Turbines," ASME Paper #90-GT-150.

Sharma, O. P., Ni, R. H. and Tanrikut, S., "Unsteady Flows in turbines - Impact on Design Procedure," from *Turbomachinery Design Using CFD*, AGARD Lecture Series No. 195, given at the Ohio Aerospace Institute, May 1994.

Simon, T. W. and Volino, R. J., 1994, "Documentation of Boundary Layer Characteristics for Low Chord-Reynolds-Number Flow on the Suction Surface of a Low-Pressure Turbine Airfoil," Final Report for the Summer Faculty Research Program and the Graduate Student Research Program, Wright Laboratories, Wright Patterson AFB.

Table 1 Cases for which pressure profiles have been measured in the Minnesota facility.

		Re			
		22,000	40,000	80,000	120,000
	0.8%	X	X	X	X
TI	3.1%	X	X	X	
	10%	X	X	X	

Note: Reynolds number, Re, is based on inlet velocity
and axial chord length

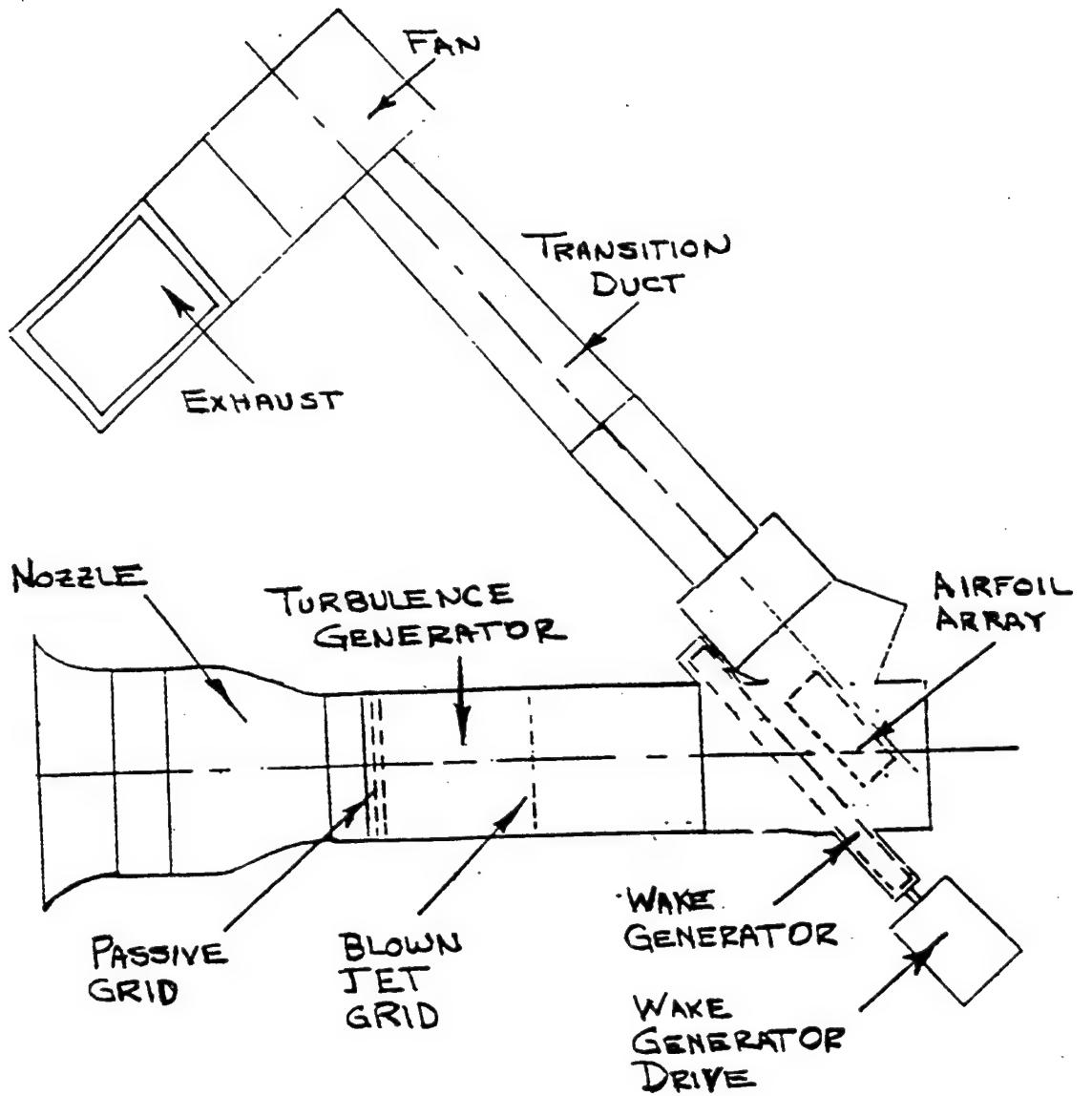


Figure 1 Plan view of the AFIT cascade tunnel at Wright Labs.

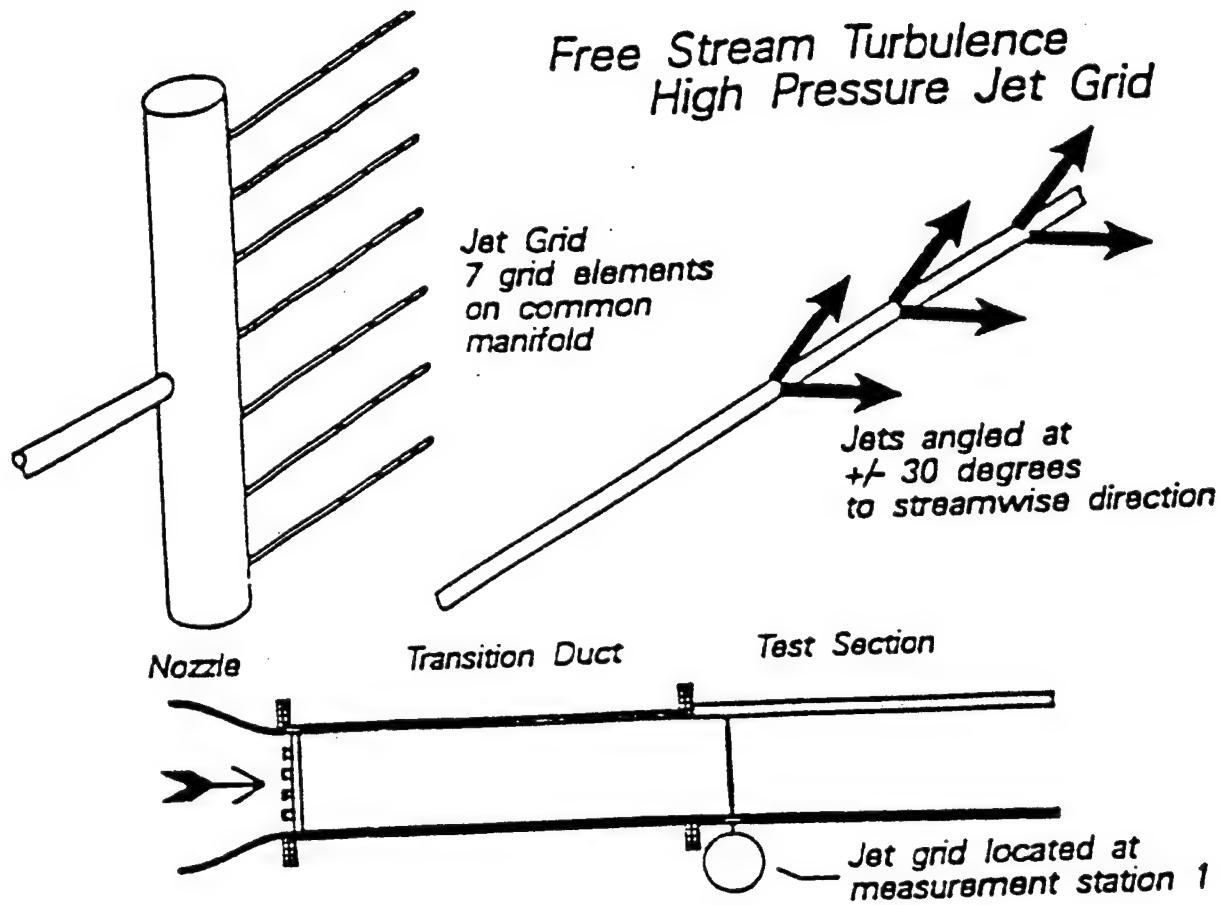


Figure 2 Schematic of the jet grid. From Sahm and Moffat, 1992.

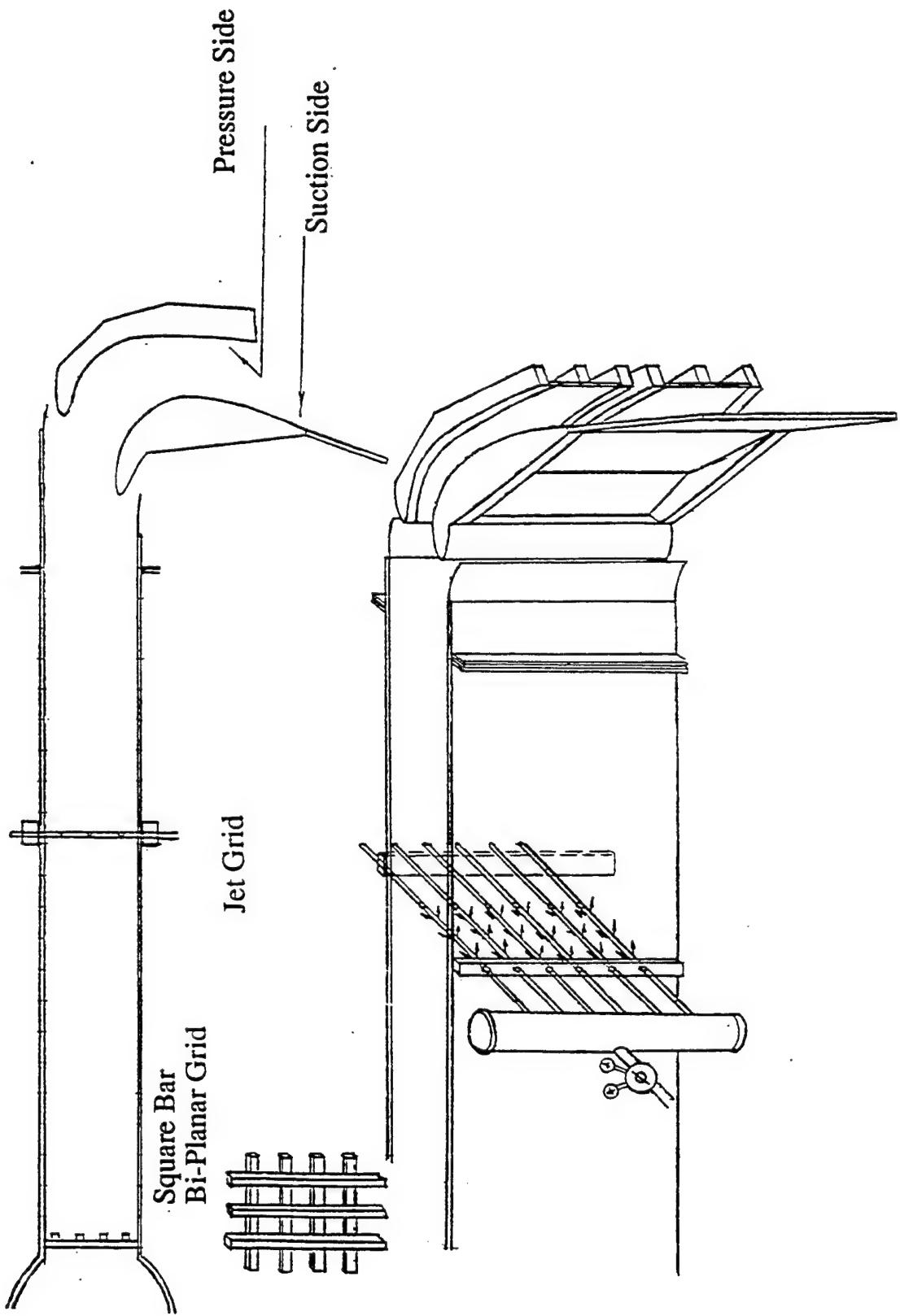
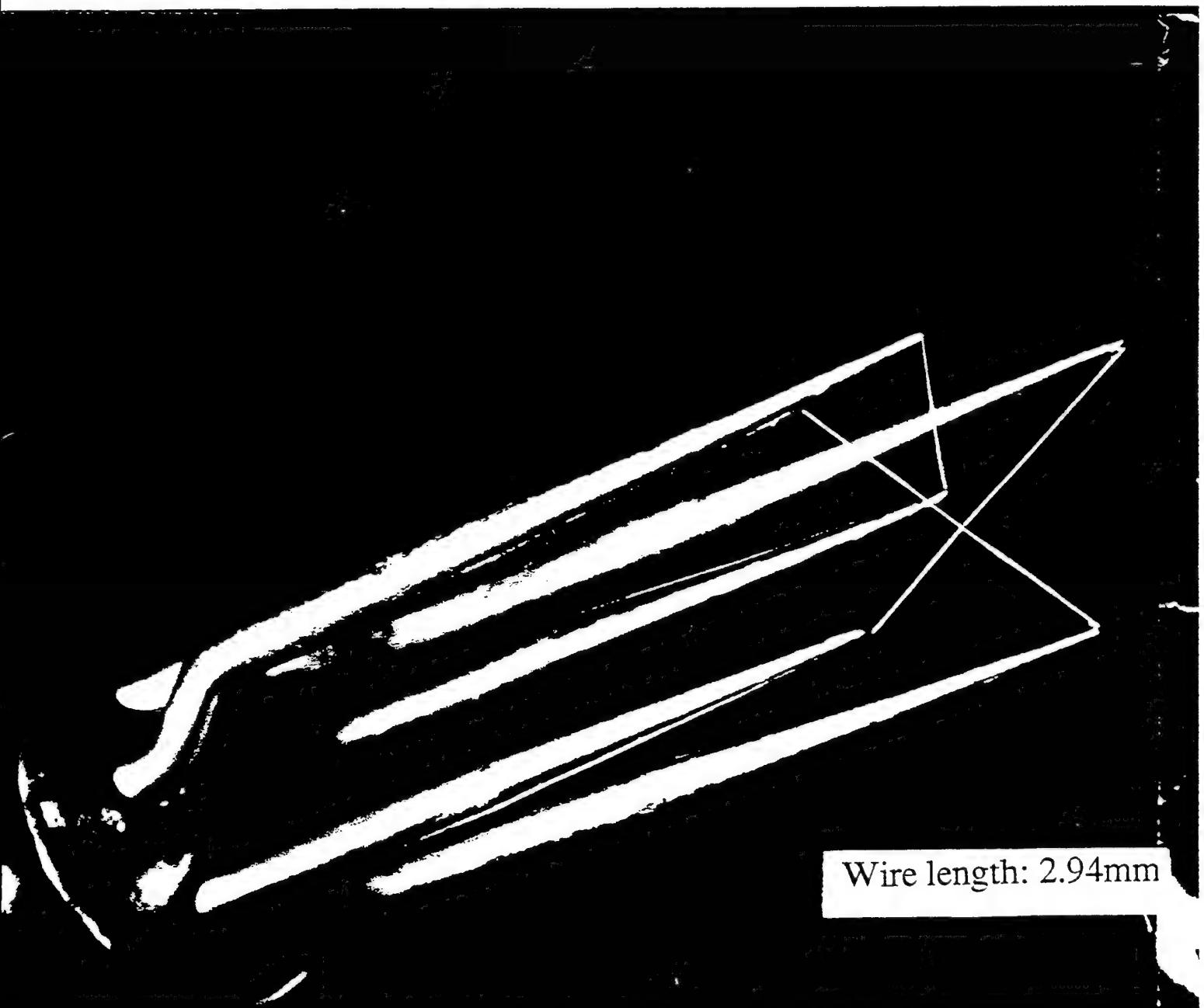


Figure 3 Turbulence generator and test section for the University of Minnesota facility.

Figure 4 Triple-wire probe for measuring three components of velocity and all components of the Reynolds stress tensor.



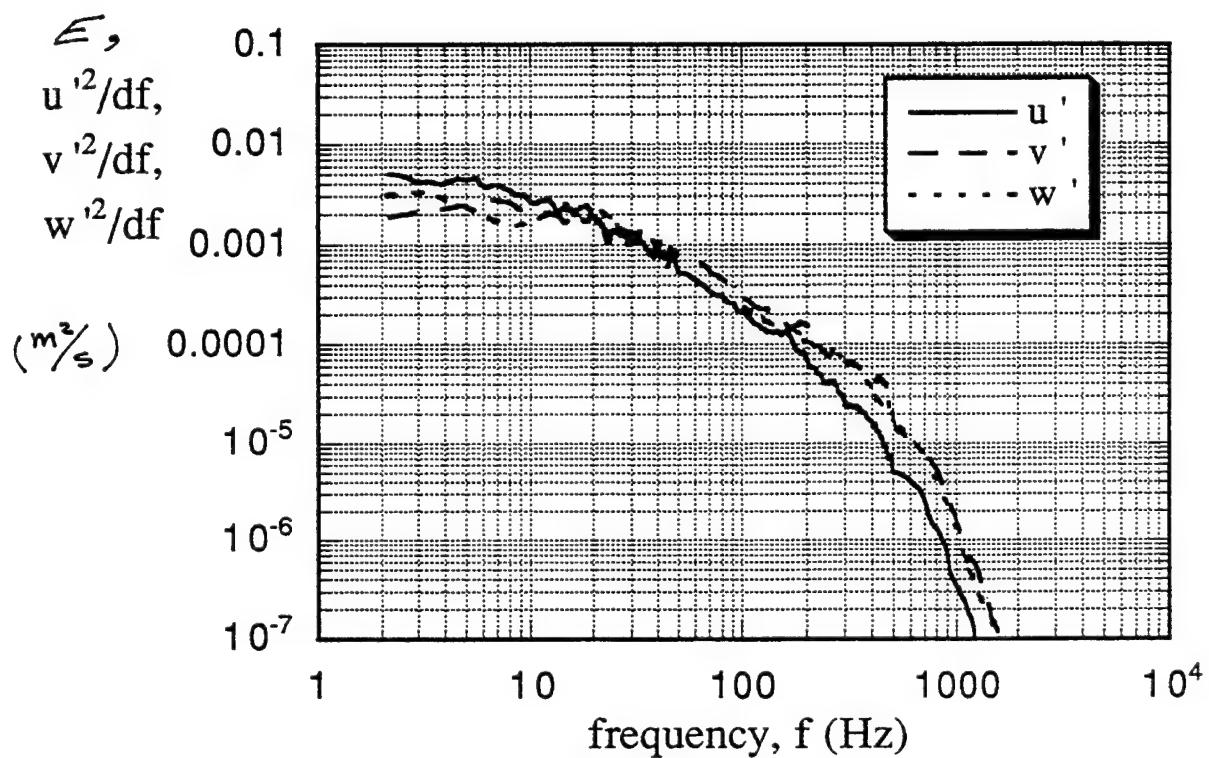


Figure 5 Spectral data taken in the approach flow of the Minnesota facility.

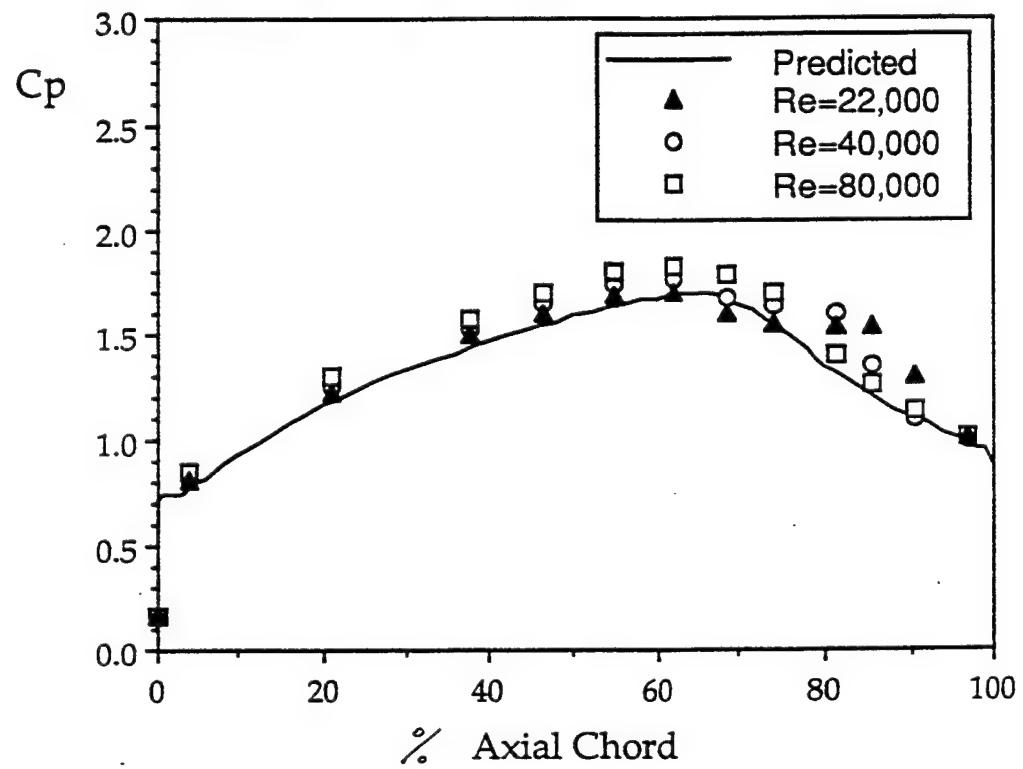


Figure 6 Pressure profiles for various chord Reynolds number values with a single free-stream turbulence intensity (3%).

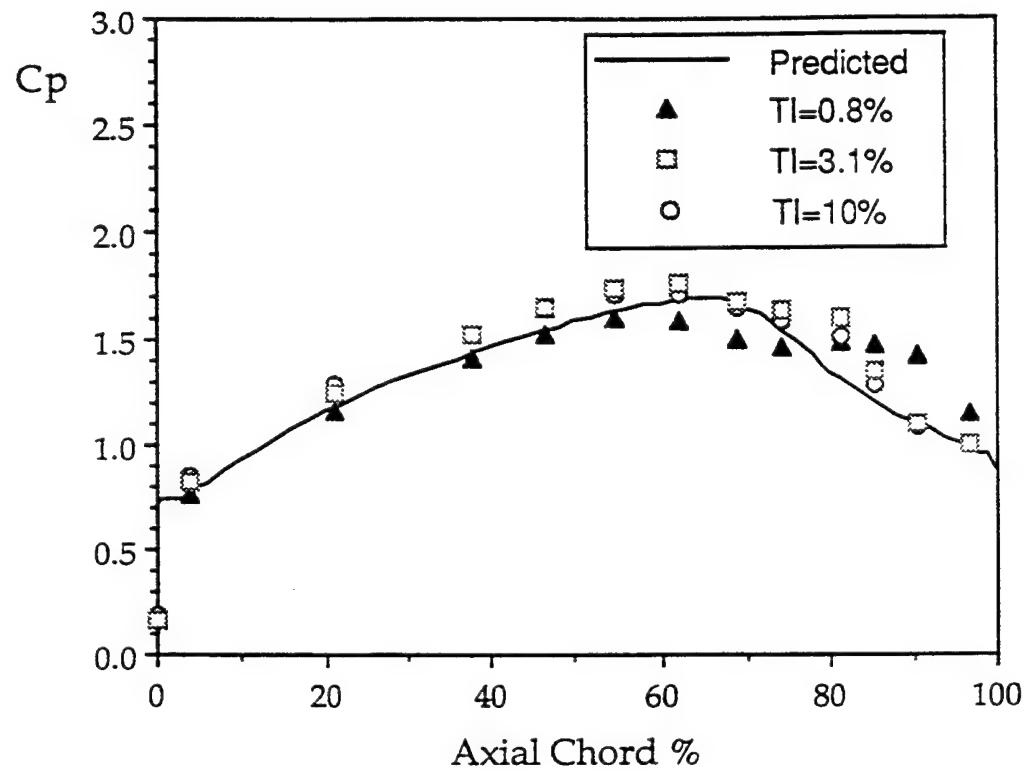


Figure 7 Pressure profiles for various free-stream turbulence intensity values with a single chord Reynolds number (40,000).

Marek Skowronski report unavailable at time of publication.

VHDL-93 Parser in SWI-Prolog

Krishnaprasad Thirunarayan
Associate Professor
Department of Computer Science and Engineering

Wright State University
Dayton, OH

Final Report for:
Summer Research Extension Program

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base
Washington, DC

and

Wright Laboratory

December 1995

VHDL-93 Parser in SWI-Prolog

Krishnaprasad Thirunarayan

Associate Professor

Department of Computer Science and Engineering

Wright State University

Abstract

The VHDL-87 parser written in Quintus Prolog by Peter Reintjes is based on a simplified grammar of IEEE Standard 1076-1987 of VHDL given by Lipsett, Schaefer and Ussery. This report is concerned with its upgrading to conform to the IEEE Standard 1076-1993 of VHDL, and subsequent porting to SWI-Prolog (Version 2.1.0) available in the public domain. The VHDL-93 parser has been used as the basis for a pretty printer, and is being used for implementing a query system to browse a VHDL-93 design description. The parser has been tested against VHDL-93 examples.

Introduction

VHDL is an acronym for VHSIC (Very High Speed Integrated Circuits) Hardware Description Language [1]. It is a “formal” language for specifying, documenting, and communicating hardware designs and models at many levels of abstraction ranging from the behavioral level to the gate level. The description can also be used to test the correctness of the circuit via simulation, or to synthesize a gate level description.

As the hardware designs get larger, the corresponding VHDL descriptions become unwieldy. To access and organize information about the hardware components of the designs (such as gates, adders etc) and the software components of the design descriptions (such as subprograms, processes etc), we need an “intelligent browser”. This can be created by assimilating the VHDL description in a suitable database, and then querying the database to obtain relevant information about the hardware and the software components.

The goal of the VHDL parser written in Prolog by Peter Reintjes [7] is to transform a VHDL program into a Prolog *term* representing its abstract syntax. Once this term is asserted into the Prolog database, simple Prolog programs can be written to access information stored in it. Furthermore, the abstract syntax tree can in fact serve as the basis for building a formatter/pretty printer.

The original VHDL-87 parser [7] written in Quintus Prolog is based on a simplified grammar of IEEE Standard 1076-1987 of VHDL [6]. This report is concerned with:

- The upgrading of the VHDL-87 parser to conform to the IEEE Standard 1076-1993 of VHDL [4].
- The porting of the VHDL-93 parser from Quintus Prolog to SWI-Prolog.
- The use of VHDL-93 parser in SWI-prolog as the basis for a design query system and a pretty printer.

The SWI-Prolog is an implementation of Prolog designed for experiments with logic programming [8]. It provides a very rich environment for program development which includes a fast compiler, tools for profiling, a history facility, and mechanisms for flexible integration with C, command-line editing, automatic completion of atom names, automatic loading of library predicates, multi-argument indexing etc. Furthermore, it is available in the public domain (in contrast with Quintus Prolog). This makes it accessible to a much larger audience, and on a wider variety of machines (including PCs).

Discussion of the Problem

We now enumerate the errors in the original parser and the extensions required to make it conform to IEEE Std 1076-1993 of VHDL. We also mention some of the problems we faced in porting the parser from Quintus Prolog to SWI-Prolog. We assume that the reader is familiar with [7] (or at least [6]) and has access to [4].

Errors in Tokenizer

The VHDL-87 tokenizer [7] does not conform to IEEE Standard 1076-1987 [3] as follows.

- In VHDL-87, the keywords and the ordinary identifiers are not case-sensitive, while strings are. The original tokenizer did not fully implement this aspect.
- The based literals (numbers in base 2 to 16) were not recognized.

- The comments at the end of a program file caused error.
- Strings and extended identifiers were permitted to contain “non-tab-format-effectors”.
- The use of ticks in VHDL-87 in different contexts, such as heralding an attribute or delimiting a character literal, poses problems to the tokenizer. For instance, constructions such as `character('a')`, `a'b'c'd`, etc. are tokenized incorrectly. (However, the interpretation of `range` as an attribute or as a keyword has been dealt with correctly.)

Extensions to Tokenizer

The IEEE Standard 1076-1993 [4] enriches the lexical structure of the language in the following ways.

- The alphabet has been extended from seven-bit ASCII to ISO eight-bit coded character set (ISO 8859-1987).
- Extended identifiers have been introduced to allow identifiers with non-alphanumeric characters. These identifiers are case-sensitive (in contrast with ordinary identifiers and keywords).

Errors in Parser

The VHDL-87 parser [7] does not conform to IEEE Standard 1076-1987 [3] as follows. (However, it must be mentioned here that the original parser explicitly assumed that the input VHDL-program is syntactically correct.)

- The original VHDL-87 parser incorrectly permitted all possible declarative items for all construct types (that can contain declarations). For instance, it permitted subprograms in configuration statement, attribute specifications in package body etc.
- The definition of `interface_element` imposes constraints on the mode, the bus-type, and the initialization expression based on the object-class. The original parser permitted all combinations.
- The original parser permits the names of entity, configuration, component, unit, function etc. to be only an identifier or a selected name. This is not

entirely correct because it does not accomodate function calls such as `Clock'Delayed'Stable(T)` etc.

- It incorrectly allows records with no fields, but omits zero-arity function calls and incomplete type declarations.
- The original parser does not enforce restrictions such as non-associativity of `nand`, `nor`, and `**`, or ensure proper use of sign in expressions like `(5 + - 5)` etc. Furthermore, it does not permit valid strings such as `(- abs 4 * 2 + 8)` etc. which contain a legal sequence of unary operators.

Extensions to Parser

The IEEE Standard 1076-1993 [4] extends the syntax of the language in the following ways. (The first three changes contribute to the *orthogonality* of the language.)

- The `end` statements are now more uniform in that they may optionally specify the construct type and the corresponding identifier/label. For example, `end entity NAME`, `end package body NAME`, `end if LABEL`, `end function NAME`, etc. are now permitted.
- The use of keyword `is` is more consistent in VHDL-93. For instance, `block NAME is`, `component NAME is`, and `process NAME is` etc. are also permitted.
- The use of labels is more consistent in VHDL-93. For instance, the `if` statement and the `case` statement can have a label now.
- VHLD-93 introduced the `report` statement that is actually a special case of the assertion statement in wide use.
- VHLD-93 introduced the keywords `pure` and `impure` in subprogram declaration, and the keyword `shared` in variable declaration.
- VHLD-93 introduced the keywords `inertial` and `reject` in delay mechanism (in signal assignment statement), the keyword `unaffected` in waveform, and the keyword `postponed` in concurrent assertion statement, concurrent procedure call statement, concurrent signal assignment, process statement etc.

- The syntax of file declarations has been changed. In fact, the VHDL-87 file declarations are no longer compatible with the VHDL-93 standard. Furthermore, the file names can now be passed as parameters to subprograms.
- The *generate* statement now has an optional declaration.
- The clause for conditional-waveform has been modified to accomodate `waveform when condition-form` (that is, an else-part is not required).
- VHDL-93 supports *group*, *group template* and *signature* constructs.
- VHDL-93 generalizes *component_name* to *instantiated_unit* in component instantiation statement.
- VHDL-93 introduces the *shift operators* and the *shift expressions*.

Differences between SWI-Prolog and Quintus Prolog

We need the following Quintus Prolog libraries for implementing the parser: `library(basics)`, `library(files)`, `library(strings)`, `library(math)`, and `library(ctypes)`. In particular, we need to import `append/3` and `member/2` from `library(basics)`, `file_exists/1` from `library(files)`, `pow/3` from `library(math)`, `concat_atom/2` from `library(strings)`, and a host of other character classification predicates from `library(ctypes)`. Furthermore, we require the built-in `peek_char/1` to tokenize the tick character in the VHDL input correctly.

SWI-Prolog supports `append/3`, `member/2` and `concat_atom/2` predicates as built-ins. `file_exists/1` predicate of Quintus Prolog is the same as the SWI-Prolog built-in `exists_file/1`, while the `pow/3` predicate can be obtained using the built-in exponentiation operator “`^`”. However, the SWI-Prolog library `ctypes` is very different from the corresponding library in Quintus Prolog. In particular, it seems to be based on the ASCII character set instead of the ISO eight-bit coded character set (ISO 8859-1987). Furthermore, SWI-Prolog does not support the built-in `peek-char/1`.

Methodology

The tokenizer and the parser has been modified to reflect the corrections to the original parser and to incorporate changes to meet the VHDL-93 standard. To

avoid unnecessary repetition we describe only a few important changes to the parser in detail here.

Changes to Tokenizer

The tokenizer code was substantially revised to overcome the problems stated above. Here, we discuss only the tick problem, in detail.

The handling of ticks in VHDL is relatively complex. This is because, ticks can occur in several different contexts that can “interact” in complicated ways, as explained below. A single tick can be used as a *separator* in a qualified expression (such as `typename'(value)`) and in attribute names (such as `prefix'attribute`). A pair of ticks can be used as *delimiters* in character literals. Given these facts, the lexical analysis of constructions such as `character('a')`, `A'B'C`, `T'range of range'a'to'b'`, and `"or"''A'B` (borrowed from internet postings of Joerg Lohse and Jacques Rouillard and [5]) becomes rather involved. However, the following disambiguation rule (adapted from Gary Beihl’s posting on the internet) can be used to guide the tokenizer:

A tick begins a character literal *if and only if* the previous token is *not* a non-keyword identifier or attribute name or a close paren or a string literal and there is a matching tick to terminate the character literal exactly two characters ahead in the input stream. In other cases, the tick can either flag a potential attribute designator or is a token by itself.

To implement this rule we need to know the previous token type and have the ability to peek one character ahead (in addition to the one lookahead we already have in the tokenizer). The VHDL-93 parser compiled for SUN-3/SUNOS-4.1 handles the “quirky” examples involving ticks satisfactorily, while the VHDL-93 parser compiled for MIPS/Ultrix and VAX/Ultrix does not. This is because, in the latter case, the underlying Quintus Prolog implementation does not support the built-in `peek_char/1`. In summary, with the additional “peeking” capability, it is possible to make the tokenizer an independent module of the parser.

Changes to Parser

The original parser code has been revised to reflect the “cosmetic” changes and new additions made to upgrade VHDL-87 to VHDL-93, as explained before. In

particular, parsing of expressions will be completely redone to enforce restrictions such as non-associativity of `nand`, `nor`, and `**`, and to ensure proper use of sign in expressions like `(- 5 + 5)`.

Overall, we observed that the performance of the VHDL-93 parser degraded somewhat on the available VHDL-87 test suite, compared to the original parser.

Porting to SWI-Prolog

As explained earlier, the following important changes were made in porting the parser code from Quintus Prolog to SWI-Prolog: Addition of the built-in predicate `peek_char/1` and the porting of the Prolog library `ctypes` designed for the eight-bit coded character set ISO 8859-1987. Furthermore, we encountered other small incompatibility problems (such as built-in `atom_chars/2` being absent from SWI-Prolog etc).

Implementing a Pretty Printer

The parser takes as input a VHDL-93 design entity and transforms it into a syntax tree. The pretty printer we have implemented takes this syntax tree and outputs a “nicely” formatted VHDL-93 program text. The formatted output improves readability by making clear the logical structure of the programs, and by imposing a “uniform look” on the programs.

Implementing a Design Query System

VHDL descriptions of nontrivial hardware circuits are large. To access information about the organization and the architecture of the hardware design, we create a Prolog database from the VHDL description, and then query it using simple Prolog programs. For example, the syntax tree constructed by the parser can be assimilated into Prolog, and queried to verify or to determine the type and the number of subcomponents of a component. The same technique can also be used to obtain relevant information about the software components. Ms. Laura Debrock, who is a graduate student in the Department of Computer Science at Wright State University, is in the process of implementing such a design query system as a part of her Master’s thesis [2]. Furthermore, we also plan to investigate building a graphical user interface based on TCL/TK to make the system easy to use. In particular, we are looking at modifying the code distributed with BinProlog to create an interface.

Public Access to the System via Internet

The appendix that follows describes the detailed design and implementation of the VHDL-93 parser and pretty-printer in SWI-Prolog. The design document and the Prolog code is available for anonymous FTP from Wright State University on the host *ftp.cs.wright.edu* in the directory *pub/vhdl* as compressed files: *VHDL93_SWI.tar.Z* and *VHDL93_SWI_README*.

References

- [1] Bhasker, J., *A VHDL Primer*, Prentice Hall, Inc., 1992.
- [2] Debrick, L., *A Prolog-based Query System for a VHDL-93 Design Database*, Department of Computer Science and Engineering, Wright State University, M. S. Thesis, 1996. (forthcoming)
- [3] *IEEE Standard VHDL Language Reference Manual, Standard 1076-1987*, IEEE, NY, 1988.
- [4] *IEEE Standard VHDL Language Reference Manual, Standard 1076-1993*, IEEE, NY, 1993.
- [5] Levia, O., Maginot, S., and Rouillard, J., "Lessons in Language Design: Cost/Benefit Analysis of VHDL Features," 31st Design Automation Conference, pp. 447-453, 1994.
- [6] Lipsett, R., Schaefer, C., and Ussery, C., *VHDL: Hardware description and design*, Boston: Kluwer Academic Publishers, 1989.
- [7] Reintjes, P., "VHDL Parser in Prolog", Technical Report, Microelectronics Center of North Carolina, Research Triangle Park, 1990.
- [8] Wilemaker, J., *SWI-Prolog 1.8: Reference Manual*, University of Amsterdam, 1993.

APPENDIX: A VHDL-93 Parser and Pretty-Printer in SWI-Prolog

Krishnprasad Thirunarayan

Department of Computer Science and Engineering

Wright State University, Dayton, Ohio-45435.

tkprasad@cs.wright.edu

Peter B. Reintjes

IBM T. J. Watson Research Center

Yorktown Heights, New York-10598.

Robert L. Ewing

Wright Labs, Microelectronics Division (WL/ELED)

Wright Patterson Air Force Base, Ohio-45433.

rewing@el.wpafb.af.mil

1.1 Introduction

This technical report describes the implementation of a VHDL-93 Parser in Prolog. The original VHDL-87 parser written in Quintus Prolog and the report are due to Peter Reintjes (then at Microelectronics Center of North Carolina, Research Triangle Park). This parser has been revised to conform to the IEEE 1076-1993 standard of VHDL and ported to SWI-Prolog by the first author. A machine readable version of this parser is also available from the author upon request.

The SWI-Prolog is an implementation of Prolog designed for experiments with logic programming. It provides a very rich environment for program development and is available in the public domain. (It can be downloaded from "swi.psy.uva.nl:/pub/SWI-Prolog" using anonymous FTP).

The entire VHDL-93 parser is implemented in a little over 2000 lines (about 1000 clauses) of

The entire VHDL-93 parser is implemented in a little over 2000 lines (about 1000 clauses) of Prolog, thus making it between one tenth and one fifth the size of other VHDL implementations. This demonstrates vividly that complex VLSI systems will not require millions of lines of code if they are described at an appropriate level of abstraction.

This parser can become the basis of a wide variety of tools which require input in the VHDL language. In particular, the original VHDL-87 parser was developed in conjunction with software components for schematic entry, logic synthesis, and a universal hardware description language translator. Both VHDL-93 parser and the original VHDL-87 parser make no semantic consistency checks on the VHDL that they read, and only require that the VHDL program be syntactically correct. However, these parsers do detect both lexical and syntax errors. Furthermore, the VHDL-93 parser generates reasonable messages for lexical errors.

This implementation will also be useful to VLSI/CAD developers who are not using Prolog because this implementation of the VHDL grammar contains corrections and clarifications that other textual specifications have omitted.

For example, a direct implementation of the grammar given in *VHDL: Hardware Description and Design*, (Kluwer Academic Press, 1989) (Authors: Lipsett, Schaefer and Ussery) would contain:

- Left-recursive rules which would require special code to guard against infinite recursion, and even then could increase the running time exponentially.
- Overlapping rules, requiring arbitrarily large look-ahead buffering.
- Incorrect grammar rules.

Since some readers of a grammar specification will be charged with the task of implementing a parser, it would be nice if they could trust the specification. In the original VHDL-87 parser, all left-recursive rules have been translated into right-recursions and overlapping rules have been combined to eliminate look-ahead. However, the original parser does not recognize based literals and does not handle ticks (') correctly. Furthermore, it vastly improves the parsing of operators, though it is not entirely "correct". The current parser attempts to remedy these problems, in order to conform to the VHDL-93. But most importantly, this "specification" in Prolog is actually an executable program which has successfully parsed a number of VHDL files, an exercise which flushed out errors in the grammar rules.

Programming Methodology

This document was produced using the \TeX typesetting system. These \TeX source-files also contain the Prolog code for the VHDL-93 parser. Running the filter program *texcode* (included in the distribution) on these files will produce the Prolog source files. If modifications are made to this parser, they should be made to the \TeX files, thus preserving the integrity of the relationship between the documentation and the code.

Data Structures

The parser builds a simple parse-tree reflecting the structure of the VHDL grammar rules. Once in this internal form, this description can be simulated, translated into other high-level circuit description languages, or interrogated by Prolog queries.

Since there are many optional constructs in VHDL, the naive parse will frequently contain the atom *null* when an optional structure is missing. Alternatively, when the optional item is a list of objects, a simple nil list (`[]`) may appear. The parse tree would be much simpler if missing optional items caused the creation of slightly different terms. For example, if *term/5*

has three optional sub-units which are frequently missing, it might be worthwhile to substitute a new version, `term2`/2 when all three are missing. Thus, frequent occurrences of:

```
term(null,null,Value,Name,[])
```

could be replaced by:

```
term(Value,Name)
```

The current implementation maintains the verbose version of the parse tree for the following reasons:

- The VHDL grammar rules are simpler if only one form of term is constructed. Rules for large structures can ignore whether or not sub-structures are being constructed.
- If the optimization described above is made, parse tree transformations must contain additional rules to handle the multiple forms of each data structure.
- Finally, if needed, an optimizing filter can be easily built as shown below.

```
optimize([],[]).  
optimize([H|T],[OH|OT]) :- optimize(H,OH), optimize(T,NT).  
  
optimize(term(null,null,Value,Name,[]),term(Value,Name)).  
optimize(term2(null,null,X),term2(X)).  
.  
.  
.  
.  
.  
optimize(Tree,New) :-  
    Tree =.. [F|Args],  
    optimize(Args,0Args),  
    New =.. [F|0Args].
```

The VHDL Grammar

The original VHDL-87 parser written by Peter Reintjes is based upon the grammar in the appendix of *VHDL: Hardware Description and Design*, (Kluwer Academic Press, 1989) (Authors: Lipsett, Schaefer and Ussery). According to the authors, this grammar is simpler than the official definition in the IEEE Standard, but is functionally equivalent. Wherever there has been a departure, we have tried to patch things up. The DCG rules in the original VHDL-87 report are numbered to coincide with the grammar rules in the book. Here too, we maintain the old numbering scheme, except that the new rules have been given a rule number followed by a letter tag to indicate the point of insertion.

1.2 Reading this Document

A full appreciation of this report requires at least a reading knowledge of Prolog, although most of the grammar rules are only slightly more complex than a standard BNF description. In particular, a casual reader should probably not waste time on the section describing the tokenizer.

The names of predicates and terms are defined by the principle functor followed by a slash and the arity (number of arguments). Thus, the term `name(A,B)` is described as: `name/2`. The names of Definite Clause Grammar (DCG) rules, which are transformed into Prolog predicates with additional arguments, are given names in the form: `vhdl_prefix//1`, alluding to the

fact that the actual arity of the resulting predicate is different from the DCG arity. Thus, the name `vhdl_prefix//1` describes a DCG rule with arity equal to one, which translates into the Prolog predicate `vhdl_prefix/3`, which has an arity of three.

1.3 Calling the Parser

The goal `vhdl_read('Filename')` directs the parser to read an VHDL description from the file named `Filename.vhdl` or `Filename.vhd` and store it in the database as a `design_unit/2` term. This top-level goal to read an VHDL file is given below.

```
vhdl_read(Name) :-  
    file_path(Name,File),  
    exists_file(File),  
    see(File),  
    vhdl_get_token_line(Tokens),  
    vhdl_design_units(Tokens),  
    seen.
```

There can be any number of design units in a file. The following predicate will terminate when it successfully reads an empty token line. This will occur when we have reached the end of the file. This version prints an asterisk on standard output after reading each design unit.

```
vhdl_design_units([])      :- !.  
vhdl_design_units(Tokens) :-  
    vhdl_design_unit(Design,Tokens,[]),  
    !, write('*'), ttyflush,  
    assert(Design),  
    vhdl_get_token_line(Next),  
    vhdl_design_units(Next).
```

The predicate `vhdl_get_token_line/1` reads in tokens up to a semicolon. The list of tokens it returns does not include the semi-colon itself, so it does not appear in any grammar rules.

The rest of this document is a bottom-up tour of the parser, beginning with the tokenizer. Normally, a tokenizer is uninteresting, but it is helpful to be familiar with the primitive tokens before reading the grammar rules.

The Tokenizer

2.1 Lexical Analysis

Prolog clauses are a good way to represent finite-state machines, which in turn are the best way to construct lexical analysers or tokenizers. The basic strategy used here is to read a character and use it as the first argument to a predicate which has clauses for each particular case of interest. First-argument indexing causes this to operate with the efficiency of a case statement. Since nearly all commercial Prologs have first argument indexing, we can practically consider it to be part of the language.

VHDL-87 is based on ISO seven-bit coded character set (ISO 646-1983), while VHDL-93 is based on ISO eight-bit coded character set (ISO 8859-1987). So the Prolog predicate `get/1` that returns a non-layout character needs to be redefined as `g_get/1` as follows. (The ISO character set processing predicates used here are defined in the Quintus Prolog library called `ctypes`.)

```
% g_get(-Char) returns the next graphic (non-control, non-blank)
% character on the input. If end_of_file then return -1.

g_get(Char) :-
    repeat, get0(Char), (is_graph(Char) ; is_endfile(Char)), !.
```

One of the unpleasant things about VHDL is that while keywords and (ordinary) identifiers are case-insensitive, all other strings must have their case preserved. Unfortunately, this means that not only do `BEGIN` and `begin` have to be recognized as the same token, but so do `Begin`, `BeGin`, `begIN`, `begin`, and all other permutations. So we introduce additional predicates `c_get0/1` to normalize letters to their lower case equivalents.

```
% c_get0(-Char) returns the next character on the input converting
% an upper case letter to the corresponding lower case one.

c_get0(CharLow) :-
    get0(Char), to_lower(Char, CharLow).
```

2.2 Representing Characters

We will define the tokenizer from the bottom-up, starting with the test for the space characters and the format effectors.

```
blank(32). /* Space */
blank(160). /* NBSP */

format_effector(9). /* Horizontal Tab */
format_effector(X) :- non_tab_format_effector(X). /* others */

non_tab_format_effector(10). /* Line Feed */
non_tab_format_effector(11). /* Vertical Tab */
non_tab_format_effector(12). /* Form Feed */
non_tab_format_effector(13). /* Carriage Return */
```

These characters are defined by the decimal value of their ISO(ASCII) codes. For all other characters, we will use the "zero-quote" notation, which allows us to type the characters as they normally appear.

In Quintus Prolog, an integer followed immediately by a single quote directs the Prolog parser to interpret the *next* string as a number in that base. Thus, 2'1010 is equivalent to the decimal number ten, and 16'0f is the decimal number fifteen. As an extension to this syntax, a preceding integer of zero tells the system that the next object is a character and we want the value of its ISO(ASCII) code. This would not have worked well in the case above for the space and the format effector characters, since they are invisible in most text editors. For that reason we use the ISO(ASCII) codes 32, 9, 10 etc. For the other characters, we will use the more readable form 0'a, 0'b, 0'* etc. However, if we are not dealing with a single character, but want a list of characters, we can do better than [0'a, 0'b, 0'c, 0'd] by typing "abcd". There is no special string datatype in Quintus Prolog, so this quoted string of characters is just another way of representing a list of ISO(ASCII) codes.

2.3 How the Tokenizer Works

The token recognizer is a "look-ahead" predicate *vhdl_get_token/3* that expects the current character as its first argument. This results in a direct call to the appropriate clause (based on the character value) in Prologs which use first argument indexing. In particular, the predicate *vhdl_get_token(+LookAheadCh, -Token, -NextCh)* is always invoked with *LookAheadCh* instantiated with a character that can start a token, and it *always* returns the token starting with *LookAheadCh* in *Token* and the character following the token in *NextCh*. To be precise, *LookAheadCh* can be instantiated with a (non-blank) graphic character *except* the characters in { '#', '\$', '?', '@', '^', '_', ',', '{', '}', '~' } that can never start a valid token and the characters in { '-', ',', ';' } that need special treatment as explained later (see clauses for *vhdl_get_token_line/3*).

Each token or lexical element is either a delimiter, an operator, an identifier (which may be a reserved word), an abstract literal (number), a character literal, a string literal, a bit string literal, or a comment. The VHDL-93 (simple and compound) operators are handled by the next set of clauses, only two of which are shown explicitly here.

```
vhdl_get_token(0'<, T, NC) :- !, get0(C2), vhdl_operator(0'<, C2, T, NC).
vhdl_get_token(0'=, T, NC) :- !, get0(C2), vhdl_operator(0'=, C2, T, NC).
```

.

.

.

Note that the minus sign '-' in VHDL-93 can begin a comment, or an (unary or binary) operator. We handle '-' separately because the comment string is stripped off by the tokenizer, while the operator contributes a token to the parser.

Leading digits indicate that we are parsing an abstract literal. The abstract literals come in two flavors — the decimal literals (numbers in base 10) and the based literals (numbers in base 2 to 16). Note that the sign and the magnitude of a number are recognized as two separate tokens. Irrespective of whether the abstract literal represents a real number or an integer, the token returned is *number(Value)*. In particular, no check has been made to enforce positive exponents for integers.

```
%vhdl_abs_lit(+Base_or_first_digit,+Digit_or_delimiter,-Token,-Next_Ch)

vhdl_get_token(0'0,T,C) :- !, get0(C2), vhdl_get_number(C2,R,[0'0|R],T,C).
vhdl_get_token(0'1,T,C) :- !, get0(C2), vhdl_abs_lit(1,C2,T,C).
vhdl_get_token(0'2,T,C) :- !, get0(C2), vhdl_abs_lit(2,C2,T,C).

.
.
.
```

The code for *vhdl_get_number/5*, *vhdl_abs_lit/4* and other predicates for recognizing abstract literals (both decimal and based literals) is not shown in detail because while it is complex, it is also rather uninteresting. Interested persons are encouraged to look at the source code. Note that the original VHDL-87 parser recognized only decimal literals, and did not handle based literals.

And now, we recognize the remaining punctuation characters. These are the single characters which are not considered as operators (though they could have been handled by that rule). Note that we do not need to consider ';', ''' and blank characters because of the precondition on *vhdl_get_token* call (satisfied by the invocation context).

```
vhdl_get_token(0',',',',C) :- !, g_get(C).
vhdl_get_token(0'(',',',C) :- !, g_get(C).
vhdl_get_token(0')',',',C) :- !, g_get(C).
vhdl_get_token(0'[,',',',C) :- !, g_get(C).
vhdl_get_token(0']',',',C) :- !, g_get(C).
```

The string literals and the VHDL-93 extended identifiers are recognized as follows. (The original VHDL-87 parser did not recognize strings with embedded double quotes.)

```
vhdl_get_token(0'"',string(Cs),NC) :-
    !, get0(C), read_to_next_double_quote(C,Cs,NC).

vhdl_get_token(0'\_,identifier(T),NC) :-
    !, get0(C), read_to_next_backslash(C,Cs,NC),
    ( Cs = [_,_|_] -> true ; lex_warning(empty_id) ),
    name(T,[0'\|Cs]).
```

Finally, the remaining clause handles all tokens which begin with an alphabetic character (including the non-ASCII alphabetic ones). With first argument indexing, the call to this predicate comes directly here if the look-ahead character is not one handled above. Note the

use of case conversion predicates to normalize the identifier names.

```
vhdl_get_token(C,T,NC) :-  
  (is_alpha(C) ->  
   ( to_lower(C,CL),  
     c_get0(C2),  
     vhdl_get_token_aux(C2,CL,T,NC) )  
  ; lex_error(illegal_char,C) ).  
  
vhdl_get_token_aux(0'"',CL,T,NC) :-  
  !, get_bit_string_lit_quotes(CL,T,NC).  
  
vhdl_get_token_aux(0'%',CL,T,NC) :-  
  !, get_bit_string_lit_percent(CL,T,NC).  
  
vhdl_get_token_aux(C2,CL,T,NC) :-  
  get_id_chars(C2,Cs,NC),  
  name(Token,[CL|Cs]),  
  T = identifier(Token).
```

All identifier tokens are temporarily tagged with `identifier/1`. This tag has been introduced for efficient branching and correct recognition of character literals and attributes as explained later. This tag is stripped off when the token is passed to the parser.

Now, we define the `get_id_chars/3` predicate to read an identifier after the first character has been read. Note that both reserved words and ordinary identifiers are treated similarly.

```
% is_csym/1 is true of Ch if Ch is a letter, digit or underscore.  
% successive underscores permitted (language design issue)
```

```
get_id_chars(C,Cs,NC) :-  
  is_csym(C) ->  
  ( c_get0(C2), get_id_chars(C2,CCs,NC), Cs = [C|CCs] )  
  ; Cs = [], NC = C.
```

The bit string literals are recognized as follows.

```
get_bit_string_lit_quotes(Base,bit_string(Val),NC) :-  
  get0(C), read_to_next_double_quote(C,BinStr,NC),  
  vhdl_scan_constant(BinStr,Base,Val).
```

```
get_bit_string_lit_percent(Base,bit_string(Val),NC) :-  
  get0(C), read_to_next_percent(C,BinStr,NC),  
  vhdl_scan_constant(BinStr,Base,Val).
```

```
vhdl_scan_constant([],_,[]).  
vhdl_scan_constant([C|Cs],Base,RV) :-  
  char_value(Base,C,StrV),  
  vhdl_scan_constant(Cs,Base,NV),  
  append(StrV,NV,RV).
```

```
char_value(0'b,C,StrV) :- binary_value(C,StrV).  
char_value(0'o,C,StrV) :- octal_value(C,StrV).  
char_value(0'x,C,StrV) :- hex_value(C,StrV).
```

```

binary_value(0'0,"0") :- !.    binary_value(0'1,"1") :- !.
binary_value(C,_) :-      lex_error(illegal_digit,C,2).

octal_value(0'0,"000") :- !.    octal_value(0'1,"001") :- !.
octal_value(0'2,"010") :- !.    octal_value(0'3,"011") :- !.
octal_value(0'4,"100") :- !.    octal_value(0'5,"101") :- !.
octal_value(0'6,"110") :- !.    octal_value(0'7,"111") :- !.
octal_value(C,_) :-      lex_error(illegal_digit,C,8).

hex_value(0'0,"0000") :- !.    hex_value(0'1,"0001") :- !.
hex_value(0'2,"0010") :- !.    hex_value(0'3,"0011") :- !.
hex_value(0'4,"0100") :- !.    hex_value(0'5,"0101") :- !.
hex_value(0'6,"0110") :- !.    hex_value(0'7,"0111") :- !.
hex_value(0'8,"1000") :- !.    hex_value(0'9,"1001") :- !.
hex_value(0'A,"1010") :- !.    hex_value(0'B,"1011") :- !.
hex_value(0'C,"1100") :- !.    hex_value(0'D,"1101") :- !.
hex_value(0'E,"1110") :- !.    hex_value(0'F,"1111") :- !.
hex_value(0'a,"1010") :- !.    hex_value(0'b,"1011") :- !.
hex_value(0'c,"1100") :- !.    hex_value(0'd,"1101") :- !.
hex_value(0'e,"1110") :- !.    hex_value(0'f,"1111") :- !.
hex_value(C,_) :-      lex_error(illegal_digit,C,16).

```

The predicate called when the comment characters (--) are recognized consumes the rest of the current line. If this parser is used as part of a translation system, this must be changed to preserve comments so that they can be reproduced in the output description.

```

vhdl_consume(C) :- non_tab_format_effect(C), !. % end-of-line
vhdl_consume(_) :- get0(C), vhdl_consume(C).

```

The following list of reserved words is used to generate a set of facts representing tables

vhdl_reserved/3, *vhdl_keyword/1* and *vhdl_token/1* (see appendix).

'Reserved Words'(vhdl,

```
[abs, access, after, alias, all, and,
architecture, array, assert, attribute,
begin, block, body, buffer, bus,
case, component, configuration, constant,
disconnect, downto,
else, elsif, end, entity, exit,
file, for, function,
generate, generic, group, guarded,
if, impure, in, inertial, inout, is,
label, library, linkage, literal, loop,
map, mod,
nand, new, next, nor, not, null,
of, on, open, or, others, out,
package, port, postponed, procedure,
process, pure,
range, record, register, rem, report,
return, rol, ror,
select, severity, signal,
shared, sla, sll, sra, srl, subtype,
then, to, transport, type,
unaffected, units, until, use,
variable,
wait, when, while, with,
xnor, xor,
'+', '- ', '**', '*',
'&', '|', '!',
'/', '/=', ':', ':=', '=',
'<=' , '<', '>=' , '=' , '>' , '>>',
'(', ')', ';', ',', '[', ']',
token(number), token(string), token(bit_string), token(char)]).
```

This includes four special token terms which are created by the tokenizer and must be treated as language tokens by the grammar.

Finally, here are the rules for single/double character operators recognized by the tokenizer.

```
vhdl_operator(0'* , 0'* , '**', NC) :- !, get0(NC).
vhdl_operator(0'* , C, '*', C) :- !.
vhdl_operator(0' / , 0'= , '/=', NC) :- !, get0(NC).
vhdl_operator(0' / , C, '/', C) :- !.
vhdl_operator(0': , 0'= , ':=' , NC) :- !, get0(NC).
vhdl_operator(0': , C, ':', C) :- !.
vhdl_operator(0' < , 0'= , '<=' , NC) :- !, get0(NC).
vhdl_operator(0' < , 0'> , '>>' , NC) :- !, get0(NC).
vhdl_operator(0' < , C, '<' , C) :- !.
vhdl_operator(0' > , 0'= , '>=' , NC) :- !, get0(NC).
vhdl_operator(0' > , C, '>' , C) :- !.
vhdl_operator(0'= , 0'> , '=' , NC) :- !, get0(NC).
vhdl_operator(0'= , 0' < , '=' , NC) :- !, get0(NC).
vhdl_operator(0'= , C, '=' , C).
% clauses for +, -, &, and | not needed
```

2.4 Reading Lines of Tokens

Finally, we come a full circle to the predicate mentioned in the first section, the token line reader. The `vhdl_get_token_line/1` predicate will read tokens up to a semi-colon, or end-of-file. This predicate calls `vhdl_get_token_line/3` with a blank look-ahead character. The mode argument required for appropriate handling of ticks is explained later.

```
vhdl_get_token_line(Line) :-  
    vhdl_get_token_line(0', charlit_mode, Line).
```

The `vhdl_get_token_line/3` predicate does the actual work by indexing on the look-ahead character when it is either a semicolon, a blank, a dash-minus, a tick, the end-of-file indicator, or a normal character. Basically, `vhdl_get_token_line/3` reads over all the characters on the input that *do not* contribute to a token (such as blanks, comments etc), and then invokes `vhdl_get_token/3` to read a token. The dash-minus and the tick need special treatment. The characters { '#', '\$', '.', '?', '@', '^', '_', '"', '{', '}', '~' } can never start a valid VHDL-93 token, and so, cause error.

```
vhdl_get_token_line(-1, _, []) :- !. % end-of-file  
vhdl_get_token_line(0'; _, []) :- !. % end of VHDL statement
```

```
vhdl_get_token_line(0'-, Mode, Tokens) :-  
    !, get0(NC),  
    ( (NC == 0'-) :-> (get0(Ch), vhdl_consume(Ch), get0(Next),  
        vhdl_get_token_line(Next, Mode, Tokens) )  
    ; (vhdl_get_token_line(NC, charlit_mode, Ts),  
        Tokens = [ '-' / Ts ] )  
    ).
```

This clause handles comments correctly. The original VHDL-87 parser would give an error if the program file ended with a comment because it did not treat this case separately.

The handling of ticks in Ada-based languages is relatively complex. This is because ticks can occur in several different contexts that can “interact” in complicated ways as explained below. A single tick can be used as a *separator* in a qualified expression (such as `typename'(value)`), and in attribute names (such as `prefix'attribute`). A pair of ticks can be used as *delimiters* in character literals. Given these facts, the lexical analysis of constructions such as `character('a')`, `A'B'C`, `T'range of range'a' to 'b'`, and `"or"''A'B` (borrowed from internet postings of Joerg Lohse and Jacques Rouillard) becomes rather involved. However, the following disambiguation rule (adapted from Gary Beihl's posting on the internet) can be used to guide the tokenizer:

A tick begins a character literal if and only if the previous token is not a non-keyword identifier or attribute name or a close paren or a string literal and there is a matching tick to terminate the character literal exactly two characters ahead in the input stream. In other cases, the tick can either flag a potential attribute designator or is a token by itself.

To implement this rule, we need to know the previous token type and have the ability to peek one character ahead (in addition to the one lookahead we already have in the tokenizer). The information about the previous token is captured by the mode argument. In particular, `vhdl_get_token_line/3` “operates” in one of two — `charlit_mode` or `attr_mode` — depending on whether or not it should expect a character literal after a tick. In summary, with the additional “peeking” capability, it is possible to make the tokenizer (lexical analyzer) an independent module of the parser.

In this report, we have not shown the code for handling tricky cases involving ticks, but the interested reader can peek at the source code for the details.

```

vhdl_get_token_line(0'',charlit_mode,Tokens) :-  

    !, get0(C1),  

    (is_print(C1) ->  

        peek_char(C2),  

        (C2 == 0'' ->  

            ( get0(C2), get0(NC),  

              vhdl_get_token_line(NC,charlit_mode,Ts),  

              Tokens = [char(C1)|Ts] )  

        ; lex_error(unmatched_tick,C1) )  

    ; lex_error(illegal_char_after_tick,C1) ).  
  

vhdl_get_token_line(0'',attr_mode,Tokens) :-  

    !, get0(C),  

    vhdl_get_token_line_after_tick(C,Tokens).  
  

vhdl_get_token_line(Blank,Mode,Tokens) :-  

    is_space(Blank),  

    !, g_get(C),  

    vhdl_get_token_line(C,Mode,Tokens).  
  

vhdl_get_token_line(C,_,[Token|Ts]) :-  

    vhdl_get_token(C,T,Next),  

    strip_id_tag(T,Token),  

    (special_tick_token(T) ->  

        vhdl_get_token_line(Next,attr_mode,Ts)  

    ; vhdl_get_token_line(Next,charlit_mode,Ts) ).  
  

strip_id_tag(identifier(Name),Name) :- !.  

strip_id_tag(T,T).  
  

special_tick_token(identifier(Name)) :- \+ vhdl_keyword(Name).  

special_tick_token(')').  

special_tick_token(string(_)).  

special_tick_token(attr(_)).
```

2.5 Summary of Changes

We now highlight the revisions made to the original VHDL-87 tokenizer written in Quintus Prolog to conform to VHDL-93 standard.

- The alphabet has been extended from seven-bit ASCII to ISO eight-bit coded character set (ISO 8859-1987).
- The keywords and ordinary identifiers are not case sensitive, while extended identifiers and all other strings are.
- The clauses for recognizing based literals (numbers in base 2 to 16) have been added.

- The comments are handled satisfactorily in this version. In the original parser, comments at the end of a file caused error.
- Strings and extended identifiers are not permitted to contain non-tab-format-effectors.
- Replacement characters for double quotes (in strings), vertical line (delimiter) and number sign (in based literals) have been incorporated.
- The use of ticks in VHLD-93 in different contexts causes problems for the tokenizer. The tick problem has been addressed.
- Error/warning messages have been added.
- Quintus Prolog library predicates have been used wherever possible. To port the parser, the source code of the library predicates `append/3`, `member/2`, `file_exists/1`, `concat_atom/2`, `pow/3` and `library(ctypes)` may be required.
- The token returned by the original VHDL-87 tokenizer for the bit string literals has the form `bit_string(Numerical_value,Base,Length)`, while the VHDL-93 tokenizer returns `bit_string(binary_string)`.
- The VHDL-93 tokenizer is a little slower than the original VHDL-87 tokenizer.
- Both, the original VHDL-87 tokenizer and the VHDL-93 tokenizer, do not partition abstract literals into integer literals and real literals. However, the tokenizer can be extended to discriminate between the two.

2.6 Lexical Overhead

We now discuss certain elements of the lexical structure of VHDL-93 that seem to complicate the implementation without really contributing much to programmer convenience.

Identifiers and Abstract literals

In the syntax of identifiers and abstract literals (numbers), successive underscores are not permitted. The implementation can be simplified by permitting sequence of underscores without sacrificing compatibility with VHDL-87. Similarly, not allowing an underscore to start the fractional part or the exponent part of a number seems unnecessary. If the fractional part is zero, a zero need not be required to follow the period. Banning an empty extended identifier does not seem justified.

Replacement Characters

In the syntax of based literals, the two matching double quotes delimiters can be replaced by two matching percent characters. The implementation can be simplified by allowing double quotes to be matched with percent. However, this simplification will not work for strings.

In the syntax of strings, the VHDL-93 does not treat the double quotes delimiter and the percent delimiter similarly. In particular, the “reasonable” string `%...#...%` is illegal.

The Tick Problem

The use of tick as a separator in attribute names, as a token in qualified expressions, and as a delimiter in character literals, permits us to create VHDL-93 constructs that are “difficult” to tokenize. Replacing the tick with another character in attribute names and qualified expressions could have simplified matters. However, the backward compatibility will pose a problem then.

Porting to SWI-Prolog

We need the following Quintus Prolog libraries for implementing the parser: *library(basics)*, *library(files)*, *library(strings)*, *library(math)*, and *library(ctypes)*. In particular, we need to import *append/3* and *member/2* from *library(basics)*, *file_exists/1* from *library(files)*, *pow/3* from *library(math)*, *concat_atom/2* from *library(strings)*, and a host of other character classification predicates from *library(ctypes)*. Furthermore, we require the built-in *peek_char/1* to tokenize the tick character in the VHDL input correctly.

SWI-Prolog supports *append/3*, *member/2* and *concat_atom/2* predicates as built-ins. *file_exists/1* predicate of Quintus Prolog is the same as the SWI-Prolog built-in *exists_file/1*, while the *pow/3* and *atom_chars/1* predicates can be obtained using the built-in exponentiation operator " $^$ " and *name/1* respectively. SWI-Prolog also supports *library(ctypes)*. Furthermore, the user need not know the precise "location" of these predicates because SWI-Prolog supports automatic loading of library predicates. There are a few Quintus Prolog predicates such as *statistics/2*, *format/2*, *stream_position/2* etc that can be made to work with minor modification. The Quintus Prolog built-in *peek-char/1* is missing but can be defined in SWI-Prolog as follows:

```
peek_char(Ch) :- current_input(Stream),
                stream_position(Stream,Old,Old), get0(Ch),
                stream_position(Stream,_,Old).
```

The VHDL IEEE 1076 Grammar

The original VHDL-87 parser by Peter Reintjes is based upon the syntax given in the appendix of *VHDL: Hardware Description and Design*, (Kluwer Academic Press, 1989) (Authors: Lipsett, Schaefer and Ussery). This parser has been upgraded by the first author to the VHDL-93 standard as described here.

Most of the grammar rules are self explanatory, particularly since the DCG syntax is so close to the usual BNF syntax. For the most part, cuts (!) are for efficiency. When a rule contains a cut that affects the grammar in a fundamental way, it will usually be explained more fully.

We have also collected all the changes made to the parser in a separate section at the end.

4.1 Literals and Miscellaneous

The grammar rules first define such things as identifiers, constant values, and expressions involving monadic (e.g. NOT) and diadic (e.g. AND) operators. The values and expressions recognized here are used throughout the remaining grammar rules.

Rule 1

```
vhdl_designator(ID) --> vhdl_identifier(ID), !.  
vhdl_designator(op(ST)) --> vhdl_operator_symbol(ST).      %***FIX  
  
vhdl_operator_symbol(ST) --> [string(SL)],  
                           {name(ST,SL)},  
                           {vhdl_operator(ST)}.
```

Rule 2

```
vhdl_literal(null) --> [null],!.  
vhdl_literal(L)   --> [ bit_string(L) ], !.  
vhdl_literal(L)   --> [ string(L) ], !.  
vhdl_literal(L)   --> vhdl_abstract_literal(L).  
vhdl_literal(L)   --> vhdl_enumeration_literal(L).  
vhdl_literal(L)   --> vhdl_physical_literal(L).
```

Rule 3

We only have one kind of number at present which includes integers and floating point numbers.

```
vhdl_abstract_literal(F) --> [ number(F) ].
```

Rule 4

```
vhdl_enumeration_literal(char(L)) --> [ char(L) ], !.  
vhdl_enumeration_literal(ID) --> vhdl_identifier(ID).
```

Rule 5

```
vhdl_physical_literal(pl(AL, ID)) -->  
    vhdl_abstract_literal(AL), vhdl_mark(ID).
```

Rule 6

vhdl_identifier_list//1 is used where a comma-separated list of identifiers is *required*.
vhdl_identifier_list_LA//1 is used where a list of identifiers is just one possible option.
The significance of this will be clarified later.

```
vhdl_identifier(ID) --> [ID], { \+ vhdl_token(ID) }.
```

```
vhdl_identifier_list([ID|IDs]) -->  
    vhdl_identifier(ID), !,  
    vhdl_identifier_list_aux(IDs).
```

```
vhdl_identifier_list([]) -->  
    { parse_error('identifier list') }.
```

```
vhdl_identifier_list_LA([ID|IDs]) -->  
    vhdl_identifier(ID), !,  
    vhdl_identifier_list_aux(IDs).
```

```
vhdl_identifier_list_aux([ID|IDs]) -->  
    [','], !, vhdl_identifier(ID),  
    vhdl_identifier_list_aux(IDs).
```

```
vhdl_identifier_list_aux([]) --> [].
```

4.2 Design Unit

The Design Unit is the top-level construction of the parse tree, this is the data structure containing the information from an entire VHDL file.

Rule 7

```
vhdl_design_unit(design_unit(CI,Unit)) -->
    vhdl_opt_context_items(CI),
    vhdl_library_unit(Unit).
```

Rule 8

```
vhdl_library_unit(LU) --> vhdl_entity_declaration(LU).
vhdl_library_unit(LU) --> vhdl_configuration_declaration(LU).
vhdl_library_unit(LU) --> vhdl_package_declaration(LU).
vhdl_library_unit(LU) --> vhdl_architecture_body(LU).
vhdl_library_unit(LU) --> vhdl_package_body(LU).
```

Rule 9

```
vhdl_opt_context_items([CI|CIs]) -->
    vhdl_library_clause(CI),
    vhdl_get_more, !,
    vhdl_opt_context_items(CIs).

vhdl_opt_context_items([CI|CIs]) -->
    vhdl_use_clause(CI),
    vhdl_get_more, !,
    vhdl_opt_context_items(CIs).

vhdl_opt_context_items([]) --> [].
```

Whenever a structure inside a grammar rule ends with a semicolon, we must read in more tokens before we can continue. For this purpose we create two, zero-arity, DCG rules. The first, `vhdl_get_more//0` will read in a new token line if the current token list is empty. This rule will fail if the current token line has not been exhausted.

The second rule, `vhdl_opt_get_more//0` behaves similarly if it encounters an empty token list, but will simply return the current token list if it is not empty.

```
vhdl_get_more([], Next)   :- !, vhdl_get_token_line(Next).

vhdl_opt_get_more([], Next) :- !, vhdl_get_token_line(Next).
vhdl_opt_get_more(In, In).
```

Rule 10

```
vhdl_library_clause(library(IL)) -->
    [ library ], vhdl_identifier_list(IL).
```

Rule 11

```
vhdl_opt_use_clauses([UC|UCs]) -->
    vhdl_use_clause(UC), !, vhdl_opt_use_clauses(UCs).
vhdl_opt_use_clauses([]) --> [].

vhdl_use_clause(use([Name|Names])) -->
    [ use ], vhdl_selected_name(Name),
        vhdl_selected_names(Names).

vhdl_selected_names([Name|Names]) -->
    [ ',' ], !, vhdl_selected_name(Name),
        vhdl_selected_names(Names).

vhdl_selected_names([]) --> [].
```

4.3 Library Units

Rule 12

The grammar for Rule 12 is incorrect as given in the appendix of *VHDL: Hardware Design and Description*. The discussion of the entity declaration on page 31 correctly (but informally) gives the correct form, but the grammar rule in the appendix leaves out the optional declarative items.

Compared with VHDL-87, VHDL-93 *end* statements are now more uniform in that they may optionally specify the construct type and the corresponding identifier or label.

```
vhdl_entity_declaraction(entity(ID,GIL,PIL,DIs,Ss)) -->
    [ entity ], vhdl_identifier(ID), [ is ],
        vhdl_opt_generic_statement(GIL),
        vhdl_opt_port_statement(PIL),
        vhdl_opt_declarative_items(entity,DIs),
        vhdl_opt_entity_body(Ss),
    [ end ], vhdl_opt_keyword(entity),
        vhdl_opt_identifier(ID).
```

```

vhdl_opt_entity_body(Ss) -->
    [ begin ], !, vhdl_entity_concurrent_statements(Ss).
vhdl_opt_entity_body([]) --> [].

vhdl_opt_generic_statement(IL) -->
    [ generic ], !, vhdl_interface_list(IL),
    vhdl_get_more.

vhdl_opt_generic_statement(null) --> [].

vhdl_opt_port_statement(IL) -->
    [ port ], !, vhdl_interface_list(IL),
    vhdl_get_more.

vhdl_opt_port_statement(null) --> [].

```

Rule 13

```

vhdl_architecture_body(arch(ID,Entity,DIs,Ss)) -->
    [ architecture ], vhdl_identifier(ID),
    [ of ], vhdl_mark(Entity),
    [ is ], vhdl_opt_declarative_items(architecture,DIs),
    [ begin ],
    vhdl_concurrent_statements(Ss),
    [ end ], vhdl_opt_keyword(architecture),
    vhdl_opt_identifier(ID).

```

Rule 14

```

vhdl_configuration_declaration(conf(ID,Entity,DIs,Block)) -->
    [ configuration ], vhdl_identifier(ID),
    [ of ], vhdl_mark(Entity), [ is ],
    vhdl_opt_declarative_items(configuration,DIs),
    vhdl_block_configuration(Block),
    [ end ], vhdl_opt_keyword(configuration),
    vhdl_opt_identifier(ID).

```

Rule 15

```

vhdl_package_declaration(package(ID,DIs)) -->
    [ package ], vhdl_identifier(ID), [ is ],
    vhdl_opt_declarative_items(package,DIs),
    [ end ], vhdl_opt_keyword(package),
    vhdl_opt_identifier(ID).

```

Rule 16

```
vhdl_package_body(package_body(ID,DI)) -->
    [ package, body ], vhdl_identifier(ID), [ is ],
    vhdl_opt_declarative_items(package_body,DI),
    [ end ], vhdl_opt_keywords(package, body),
    vhdl_opt_identifier(ID).

vhdl_opt_identifier(ID) --> vhdl_identifier(ID), !.
vhdl_opt_identifier(_) --> [].

vhdl_opt_keyword(KID) --> [ KID ], !.
vhdl_opt_keyword(_) --> [].

vhdl_opt_keywords(KID1,KID2) --> [ KID1, KID2 ], !.
vhdl_opt_keywords(_,_) --> [].
```

The last part of this production, `vhdl_opt_identifier//1` (resp. `vhdl_opt_keyword//1//2`), is called with its argument instantiated. This will succeed not only when the identifier (resp. keyword) found in the input stream matches the one recognized at the beginning of the rule, but will also succeed if there is no identifier (resp. keyword). This is in contrast to the other optional rules which instantiate their arguments to `null`.

4.4 Declarative Items

The original VHDL-87 parser incorrectly permitted all possible declarative items for all construct types (that can contain declarations). Here we allow only those declarative items that are explicitly permitted by VHDL-93 for each construct type.

The first clause is applicable only to *configuration*. The rest of the clauses take care of all the other construct types.

Rule 17

```
vhdl_declarative_item(configuration,DI) -->
    !, ( vhdl_attribute_specification(DI)
          ; vhdl_use_clause(DI)
          ; vhdl_group_declaration(DI)
        ).

vhdl_declarative_item(_,DI) -->
    ( vhdl_type_declaration(DI)
      ; vhdl_subtype_declaration(DI)
      ; vhdl_constant_declaration(DI)
      ; vhdl_ordinary_variable_declaration(DI)
      ; vhdl_file_declaration(DI)
      ; vhdl_alias_declaration(DI)
      ; vhdl_use_clause(DI)
      ; vhdl_group_template_declaration(DI)
      ; vhdl_group_declaration(DI)
    ).
```

```

vhdl_declarative_item(C,DI) -->
    { shared_var_allowed(C) },
    vhdl_shared_variable_declaration(DI).

vhdl_declarative_item(C,DI) -->
    { C == package }
        -> vhdl_subprogram_declaration(DI)
        ;  vhdl_subprogram_declaration_or_body(DI).

vhdl_declarative_item(C,DI) -->
    ( { signal_allowed(C) },
    ( vhdl_signal_declaration(DI)
    ; vhdl_component_declaration(DI)
    ; vhdl_configuration_specification(DI)
    ; vhdl_disconnection_specification(DI)
    )
    ).
.

vhdl_declarative_item(C,DI) -->
    { C \== package_body },
    ( vhdl_attribute_declaration(DI)
    ; vhdl_attribute_specification(DI)
    ).

vhdl_opt_declarative_items(Construct,[I/Is]) -->
    vhdl_declarative_item(Construct,I),
    vhdl_get_more,      !,
    vhdl_opt_declarative_items(Construct,Is).

vhdl_opt_declarative_items(_,[]) --> [].

```

4.5 Groups

VHDL-93 supports the concept of groups and group templates not present in VHDL-87.

Rule 17a

```

vhdl_group_template_declaraction(group_template(TID,CL)) -->
    [ group ], vhdl_identifier(TID), [ is ],
    [ '(' ], vhdl_entity_class_entry_list(CL), [ ')' ].

vhdl_entity_class_entry_list([C|CL]) -->
    vhdl_entity_class_entry(C), !,
    vhdl_entity_class_entry_list_aux(CL).

vhdl_entity_class_entry_list([]) -->
    { parse_error('group entity class list') }.

vhdl_entity_class_entry_list_aux([C|CL]) -->
    [ ',' ], !, vhdl_entity_class_entry(C),
    vhdl_entity_class_entry_list_aux(CL).

vhdl_entity_class_entry_list_aux([]) --> [ ].

vhdl_entity_class_entry(entity_class(EC,One_Any)) -->
    vhdl_entity_class(EC),
    ( [ '<>' ], !, { One_Any = any }
    ; { One_Any = null } ).
```

Rule 17b

```

vhdl_group_declaration(group(ID,TID,CL)) -->
    [ group ], vhdl_identifier(ID), [ ':' ],
    vhdl_name(TID),
    [ '(' ], vhdl_group_constituents(CL), [ ')' ].

vhdl_group_constituents([L|CL]) -->
    vhdl_constituent(L),
    vhdl_group_constituents_aux(CL).

vhdl_group_constituents([]) -->
    { parse_error('group constituents') }.

vhdl_group_constituents_aux([L|CL]) -->
    [ ',' ], !, vhdl_constituent(L),
    vhdl_group_constituents_aux(CL).

vhdl_group_constituents_aux([]) --> [ ].

vhdl_constituent(L) --> vhdl_name(L).

vhdl_constituent(char(L)) --> [ char(L) ].
```

4.6 Subprograms

Here we give rules for subprogram declaration, and for both subprogram declaration and body combined. If we read a subprogram specification and the next token is not the keyword **is**, then we have just read a subprogram declaration. It would be impossible to backtrack out of the subprogram declaration and try the subprogram body rule because several token lines may have been read while getting the interface-list part of the subprogram specification.

Rule 18

```
vhdl_subprogram_declaration(sub_program(SS,null)) -->
    vhdl_subprogram_specification(_,_ ,SS).

vhdl_subprogram_declaration_or_body(sub_program(SS,Body)) -->
    vhdl_subprogram_specification(Kind,Name,SS),
    vhdl_opt_subprogram_body(Kind,Name,Body).
```

Rule 19

```
vhdl_subprogram_specification(procedure,D,sub_spec(D,IL,null,null)) -->
    [ procedure ], vhdl_designator(D),
    vhdl_opt_interface_list(IL).
vhdl_subprogram_specification(function,D,sub_spec(D,IL,TM,P)) -->
    ( [ pure ], !, { P = pure }
    ; [ impure ], !, { P = impure }
    ; [ ], { P = pure } ),
    [ function ], vhdl_designator(D),
    vhdl_opt_interface_list(IL),
    [ return ], vhdl_mark(TM).
```

Rule 20

```
vhdl_opt_subprogram_body(Kind,Name,program_body(Is,Ss)) -->
    [ is ], !,
    vhdl_opt_declarative_items(subprogram,Is),
    [ begin ],
    vhdl_sequential_statements(Ss),
    [ end ], ([Kind], ! ; []), vhdl_opt_designator(Name).

vhdl_opt_subprogram_body(_,_ ,null) --> [].

vhdl_opt_designator(D) --> vhdl_designator(D),!.
vhdl_opt_designator(_) --> [].
```

DEVELOPMENT OF QUALITATIVE PROCESS CONTROL DISCOVERY SYSTEMS
FOR POLYMER COMPOSITE AND BIOLOGICAL MATERIALS

Robert B. Trelease
Adjunct Assistant Professor
Department of Neurobiology

University of California, Los Angeles
73-235 Center for Health Sciences
Los Angeles, CA 90095

Final Report for:
Summer Research Extension Program
Wright Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Wright Laboratory

December 1995

DEVELOPMENT OF QUALITATIVE PROCESS CONTROL DISCOVERY SYSTEMS
FOR POLYMER COMPOSITE AND BIOLOGICAL MATERIALS

Robert B. Trelease
Adjunct Assistant Professor
Department of Neurobiology
University of California, Los Angeles

Abstract

Computer-based qualitative process modeling methodologies for simulation, process control, and discovery were studied and implemented in two disparate applications fields: Composite materials processing and its biological counterpart, medical treatment. In the composite materials processing work, ongoing research for a doctoral dissertation, the emphasis was on developing a specialized qualitative process control discovery system to manage the temperature- and pressure-mediated curing of polymer/fiber parts in an autoclave. This specific materials sciences project encompassed the design of a complete system for "self-learning" (discovery) of complex process heuristics and operational control of those processes. For operational efficiency in run-time environments, the overall discovery system design was separated into executable tasks based on program modules. The first task was envisionment, in which a qualitative process "map" was produced based on evaluation of experimental conditions in the context of specific knowledge base heuristics. The second task involved interaction of the envisionment with real process data in order to develop new heuristics. The third task involved building those new heuristics, new rules to be added to the existing knowledge base. The fourth task involved controlling the materials process based on knowledge base heuristics as defined by an envisionment, and the process "paths" (succession of state transitions) it encompassed. In the second application area, work was undertaken to define important process heuristics, to build models, and to deal with physiological data acquisition and analysis for hyperbaric oxygenation, a treatment in which elevated oxygen pressure is used to enhance repair of wounds and tissue damage in human patients.

DEVELOPMENT OF QUALITATIVE PROCESS CONTROL DISCOVERY SYSTEMS
FOR POLYMER COMPOSITE AND BIOLOGICAL MATERIALS

Robert B. Trellease

Introduction

Certain types of physical systems problems and their models must deal with incompletely understood processes and/or unquantifiable relationships, and the argument has been put forth that real dynamical physical systems cannot be completely described in quantitative terms [Oreskes et al., 1994]. To handle such difficult tasks, researchers have developed new computer-based reasoning methods, such as qualitative process (QP) theory [Forbus, 1984], that employ symbolic programming and cognitive sciences techniques used in artificial intelligence (AI) research [Weld and DeKleer, 1992; Uckun, 1992; DeKleer and Forbus, 1993].

Qualitative reasoning methods have been used successfully to solve physical sciences problems that are insoluble when approached with accepted quantitative techniques (e.g., computer-based mathematical modeling) [DeKleer and Weld, 1992]. Qualitative modeling and simulation have been used for testing systems theories, for diagnosing problems [Uckun, 1992], for controlling processes [LeClair and Abrams, 1989; LeClair, Abrams, and Matejka, 1989], and for developing "discovery systems" that evolve new process theories [Weld and DeKleer, 1992].

The purpose of this research project was to continue, to complete, and to expand on the study and development of knowledge-based, QP discovery systems for process control, work performed during the AFOSR Summer Faculty Research Program (1994). The principal focus for the initial project was the application of QP system discovery methods to support automated knowledge base (KB) development for autoclave curing of composite polymer materials, a project under way at Wright Laboratory, Wright-Patterson AFB. This study was performed in collaboration with the primary work of Frances L. Abrams (MLIB), under the supervision of Dr. Steve LeClair (MLIM). A secondary focus, constituting applied technology transfer, was to begin the creation of a QP modeling system to support discovery in hyperbaric oxygenation (HBO) a medical treatment method (in regular use by the Air Force Medical Corps) that is comparable in aspects of its application to the autoclave curing of composite materials. This latter project was undertaken in collaboration with Dr. (Colonel) Richard Henderson, AFMC, at the Hyperbaric Medicine facility of the 74th Medical Group, Wright-Patterson AFB (SGPH).

Methods: Modeling, Discovery Functions, and Problem Domain Knowledge Representation

All QP work was performed on Apple Macintosh computers, including PowerBook 180, CI, Quadra 840AV, 950, PowerMac 8100 and 8500 models, using TSC (Brownsville, CA), a symbolic, object-oriented programming environment composed of nested, extensible multiple language interpreters/compilers, run-time utilities, and an envisionment builder.

Comparable to a conventional AI expert system, TSC allows the definition of actors (e.g., different types of materials, substances, and cells), taxonomic and functional relationships, physical and physiological processes, predicated interactions, and behaviors in hierarchical sets of symbolic production rules. TSC also compiles the description of initial conditions and functional designs for simulated experimental trials to be run using defined actors, states, relationships and process behaviors. As a conventional expert system uses a logical "inference engine" to evaluate acquired data in the context of defined heuristics, TSC employs an envisionment builder to record and to visualize the evolution of process behaviors and physical interactions resulting from initial experimental conditions given the defined actors, states, and functional relationships. TSC modeling and discovery functions were previously extensively described in the final report of the AFOSR Summer Faculty Research Program preceding this project (Trelease, 1994). A more general description of the applied QP methodology is provided below.

In order to produce a computer-based discovery system for dealing with materials and processes, several critical functions must be implemented in software. First, a model must be created against which incoming real-world data is to be interpreted. Second, there must be data-handling functions to "encode" information from external databases or instrumentation and to convert it to a form usable with the modeling and discovery functions. Third, there must be functions that compare the data with the model predictions (expectations). Fourth, there must be rules and methods for handling expectation failures encountered when the data do not fit model predictions. Fifth, the system must encompass rules and procedures allowing the creation of new rules for extending the basic model based on acquired/analyzed data.

To date, no single, generic program or software suite has implemented full discovery system functions. Therefore, one of set of objectives for this project was to create specific extensions in TSC to implement the characteristic discovery-related behaviors described above. These methodology studies were performed in conjunction with Frances L. Abrams of Wright Laboratory, who was completing dissertation research on a QP modeling and discovery system for autoclave process control for automated composite materials curing.

Perhaps the most central component to the discovery system was the model (defined by a domain-specific heuristic KB) and extensive study, research, and development work was necessary to implement the KB for hyperbaric oxygenation, as described below. In operation, TSC used these KB process rules to build an "envisionment" showing all the possible process states and outcomes given the defined initial experimental conditions. For example, a composite curing envisionment would show the various temperature and state changes leading to desirable (satisfactorily cured part) and undesirable (burnt part) process outcomes.

In order to relate the course of "real world" processes on the "map" provided by the TSC QP envisionment, a separate task was created for running a LIST.BUILDER process. The principal function of the LIST.BUILDER heuristics was to find, sort, and list individual states/paths (from episode to episode) that the experimental system could take in achieving its final states. Simply put, this task would individuate and list the "branches" for a given envisionment's "decision tree".

Central to TSC's ability to interact with externally acquired "real world" data was the development of external data loading functions. For the composite curing application, incoming data were handled by encoders, which provided qualitative transformations and loaded the resultant information into appropriate slots of special actor objects (frames). For this purpose, an autoclave simulator (or in operation, the real device) was connected to the TSC-running computer via a serial port, and serial data encoders translated sensor (e.g., thermocouple) data into qualitative actors, state, and relationships. For the hyperbaric oxygenation application described below, the specialized analog data acquisition software was capable of outputting numerical data in ASCII text, tab-delimited (EXCEL) spreadsheet format, a file type in common use for many commercial numerical analysis and modeling programs. We therefore created a set of high-level TSC encoder actors to load tab-delimited ASCII numerical data in segments, essentially a column at a time, into a two dimensional array. This segmentation was found to be necessary due to system-imposed limitations on maximal loadable file size (64 Kb).

The STUDY.DATA task was designed to evaluate encoded process data, determine system actors, state, and relationships, then evaluate these against the list of possible process episode antecedents provided by the LIST.BUILDER. If there was no match with any specific episode in an envisionment, a separate task could spawn a new rule. This is one use of "exception" or "expectation failure" handling within TSC. For process control, it would be possible to extend this functionality in a TSC task adding separate output functions (encoders) to modify system state(s) in order to direct the process down a desired pathway (envisionment "branch") as designated by the list builder.

Finally, a RULE.BUILDER task was created to generate new KB rules (heuristics) based on the failure of the experimental system's data to follow state changes paths as defined in an envisionment. Given a lack of a match with envisionment episodes, the current experimental episode was converted into a *conjectured* rule (episode) defined by the actual antecedent and consequent actors, states and relations. Such new rules would be added to the KB, producing a new model, and a new envisionment could be generated to characterize system process behaviors as defined by the new heuristics.

Repeated cycles of envisionment, list building, data study, and rule building would thus constitute an incremental discovery process for the modeled experimental system. Figure 1 shows a flow chart depicting the TSC processes, tasks, and loops implementing such an automated discovery system for composite materials curing.

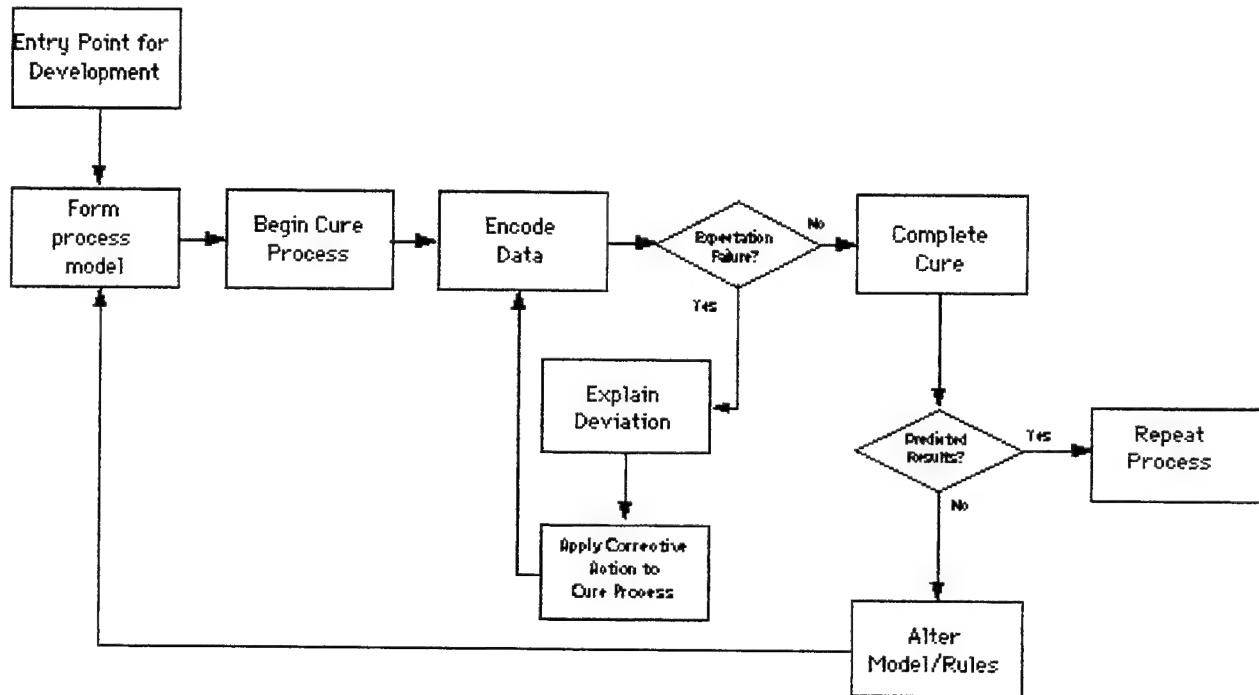


Figure 1: Flow diagram of TSC tasks and processes implementing an iterative system for rule-building (i.e., process modeling) for composite materials curing in a computer-controlled autoclave.

Polymer Composite Curing Domain Knowledge

The principal objective of the initial composite curing model was to represent the fundamental physical relationships between the fiber-polymer composite part, the controlled temperature/pressure autoclave used for curing, and part data/sensor input that could be employed for assessing cure dynamics. The most important actors in the model are shown schematically in Figure 2, and the process is described in greater detail in the following paragraphs.

Composite parts are constructed of layers of fibrous cloth-like material (parallel carbon, aramid, or glass fibers) impregnated with monomer resin. Layers of this material, called prepreg, are laminated to produce the desired part

shape/size. The alternating configuration of the fibers in prepreg layers give the finished part its characteristic strength and stiffness properties when the resin is cured (polymerized).

The laminated prepreg part is placed on a metal mold (tool) and covered with a flexible bag that is sealed over the part. A teflon sheet and matting is placed between bag and part to prevent sticking. A vacuum line attached to the bag removes the air and volatile materials which may be in or surrounding the part. This process helps to reduce voids that might form between layers in the part. The part/tool/bag assembly is referred to as a lay-up.

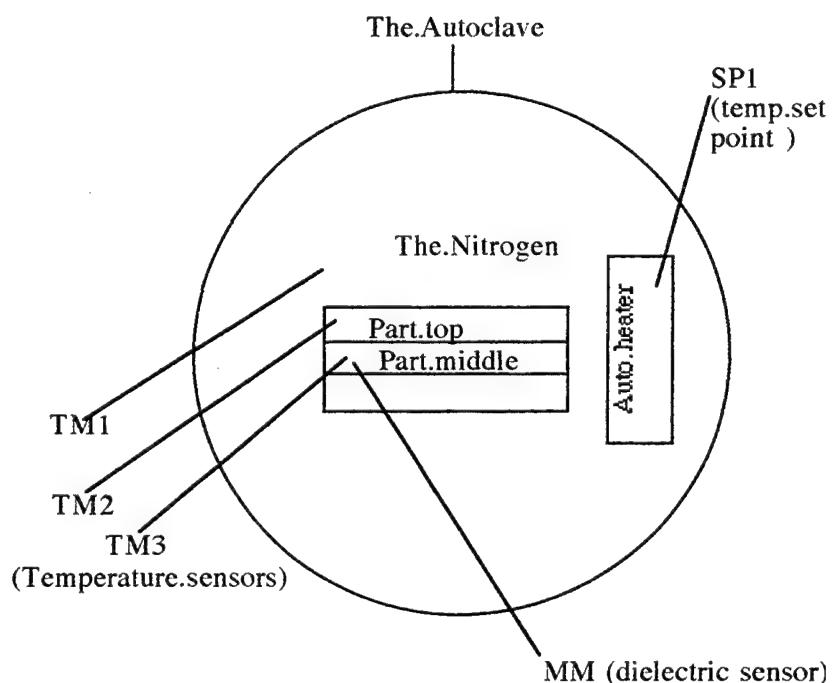


Figure 2: Diagram of the conceptually important actors in the composite materials curing model

For controlled curing, the lay-up is placed inside of the autoclave, a large pressure vessel that can be heated by circulating gas. Pressurizing the autoclave with heated gas initially lowers resin viscosity, presses fiber layers together into a continuous laminate, and provides the thermal activation energy for resin polymerization. A properly cured part is composed of fibers embedded in a solid piece of polymer without cracks or voids. If the resin is sufficiently cured, the strength and stiffness of the composite part is determined by the number and orientation of fiber layers and by the ratio of fiber to polymer (matrix).

A number of undesirable features can arise during an inadequately controlled cure cycle. As previously noted, voids within the part may be created by the formation of gas bubbles. Large temperature gradients can cause cracks or

residual stresses. Finally, since heat is generated by polymerization, there can be a runaway exothermic reaction that can burn the part.

Variables controlling the curing process include temperature and pressure of the autoclave and the magnitude of pressure or vacuum applied to the lay-up bag. Heat applied by increasing autoclave temperature can increase the degree of cure. However, a sustained temperature increase can slow the cure because of viscosity changes in the resin. Increasing pressure increases the efficiency of heat transfer within the pressure chamber, and too much or too little pressure at the wrong time can cause the formation of bubbles and voids. There are very complex, non-linear relations between autoclave temperature, pressure, bag pressure, heat transport, resin composition and viscosity, and the desired strength, temperature and chemical resistance and other characteristics of a cured part. These relationships are far too complex and poorly understood to discuss in this report. The complexity of these relationships is, in fact, the very reason why artificial intelligence (expert system) approaches have been employed in the quest to improve composite parts curing and autoclave control.

During a cure cycle, only temperature and pressure measurements can be obtained from the part and the autoclave. Pressure measurements made with diaphragm transducers are only presently reliable for the autoclave and not for the laminate/part or for the interior of the bag. Thermocouple temperature transducers will work for the part and the autoclave, and a spatial array can be used to infer rate and direction of heat transfer. For the QP model employed in this discovery system project, only three thermocouples (TM1, TM2, and TM3, as diagrammed in Figure 2) were included for assessing part cure, with differences in their temperatures and relative rates of change encoded for use in the most crucial process rules.

In the real process, autoclave heating, pressurization, lay-up pressurization/vacuum, and gas supplies are controlled via a computer serial port interface. For development of simulation and control software, an autoclave simulator was used. The autoclave simulator runs on a separate personal computer and communicates with the discovery system (TSC) host computer via a standard RS-422 serial port interface.

Hyperbaric Oxygenation Domain Knowledge

Hyperbaric oxygenation is an accepted and efficacious treatment for decompression sickness (DCS), healing of problem wounds and compromised skin grafts/flaps, mixed soft tissue infections and gangrene, burns, soft tissue and bone radiation necrosis, and carbon monoxide and smoke poisoning (Hyperbaric Oxygen Committee, 1992). Specific HBO treatment protocols for individual conditions have been empirically derived, and many of the underlying processes involved (whether cellular, physiologic, or systemic) are poorly understood or unknown.

Because work on the hyperbaric oxygenation application was begun de novo with few proven quantitative relationships established to support treatment principles, it was necessary to "start from scratch" in qualitatively modeling the physical processes involved in a wound-healing treatment, as well as the subject's (patient's) physiological responses.

Researchers have distinguished several distinct, overlapping phases of cellular and tissue responses to a wound during the healing process (Davis, 1988). As summarized in Table 1, these include an immediate (and typically brief---2 week) inflammatory phase, in which cells and various substances are mobilized in a response common to many types of tissue damage, including infection. Blood vessel permeability is increased, and immune system cells migrate into the area of damage, ready to deal with infecting organisms. This state is sustained to some degree throughout tissue repair, and it may be sustained or enhanced in certain disease states, including chronic infection. During the repair phase, immune system cells collect in the wound space (granulation), fibroblasts collect and produce new collagen (the fibrous matrix of connective tissue), blood vessels sprout and grow in, the skin surface cells regrow (re-epithelialization), and the wound volume contracts.

Table 1: Biological (Cellular and Tissue) Responses During Wound Healing

- A. The inflammatory response to wounds
 - 1. acute inflammatory reaction
 - 2. chronic inflammation
- B. The repair phase of wound healing: Re-epithelialization and contraction
 - 1. granular tissue formation
 - 2. re-epithelialization of the wound surface
 - 3. fibroplasia
 - 4. neovascularization
 - 5. wound contraction

Under normal conditions in normal individuals, these processes usually progress rapidly together, so that a moderate-sized skin wound might heal completely in less than two months. However, in the case of so-designated "problem wounds", a number of factors may interfere to inhibit or delay healing, including infections, circulatory disease, metabolic problems, immune system compromise, prior radiation treatment, and certain drugs. HBO treatment is presumed to enhance healing by providing higher oxygen levels in the affected tissues, thus alleviating the relatively low oxygen levels (hypoxia) that accompanies the primary damage and circulatory insufficiency. Furthermore, high oxygen levels have a direct toxic effect on infectious organisms, particularly anaerobic bacteria, that may flourish in certain wounds. Finally, both aspects of high-pressure oxygenation may be important in enhancing healing in

diabetic individuals, who may show circulatory problems, increased susceptibility to infection, and other disturbances of cellular functions related to tissue repair.

In HBO treatment, patients with problem wounds are placed in a pressure chamber and subjected to elevated atmospheric pressures (typically 2.4 atmospheres absolute [ATA]) for repeated periods under alternating conditions of pure oxygen (O_2) and normal air breathing. Typically, 100% O_2 breathing begins as soon as the chamber "reaches depth" (2.4 ATA), and the patient continues oxygenation for 30 minutes before beginning to breathe air again. Five oxygenation periods are separated by four 10 minute air breathing periods, then chamber repressurization takes place. This process is represented in flow diagram form in Figure 3.

In clinical practice, HBO patients are usually monitored using electrodes that measure the partial pressure of oxygen (O_2) and carbon dioxide (CO_2) passing through the skin. Electrodes are typically attached to the skin near the wound, as well as on other parts of the body (i.e., the other leg or the chest) that are presumed to have relatively normal or unimpaired circulation. With the current state of the art, such O_2 measurements are the most useful indicators of treatment efficacy, although CO_2 levels may reflect another aspect of local tissue metabolic activity (pH).

HBO treatment may produce damage or physiological disturbances referred to as oxygen toxicity, including changes in lung (pulmonary) function and seizures. The risk of oxygen toxicity places functional limits on the maximum chamber pressure, durations of oxygen-breathing and air-breathing periods, and overall duration or frequency of treatments. Thus, while it might be assumed that increasing oxygen exposure might enhance healing of particular problem wounds, the risk of damage might out-weigh the benefits of aggressive HBO treatment. HBO treatment is thus very formulaic (if empirical) and the ability of physicians and researchers to improve or tailor therapy for specific individuals or disease states is limited by the lack of understanding how all the involved processes interact in wound healing. It was deemed very appropriate, therefore, to create a QP modeling system for the purpose of better understanding HBO works to help heal wounds.

In addition to establishing important actors and processes for this unified model of HBO and wound healing, it was necessary to identify crucial data used for assessing treatment efficacy, to establish procedures for acquiring these data, to appraise functional relationships between specific data patterns and treatment outcomes for building KB heuristics, and to develop TSC functions and tasks for encoding the critical HBO data for envisionment building and discovery. Because of ethical constraints on medical experimentation and technical limitations (the obsolescent HBO chamber controller was not amenable to external computer control), however, it was not possible to create TSC process control tasks comparable to those for composite materials curing.

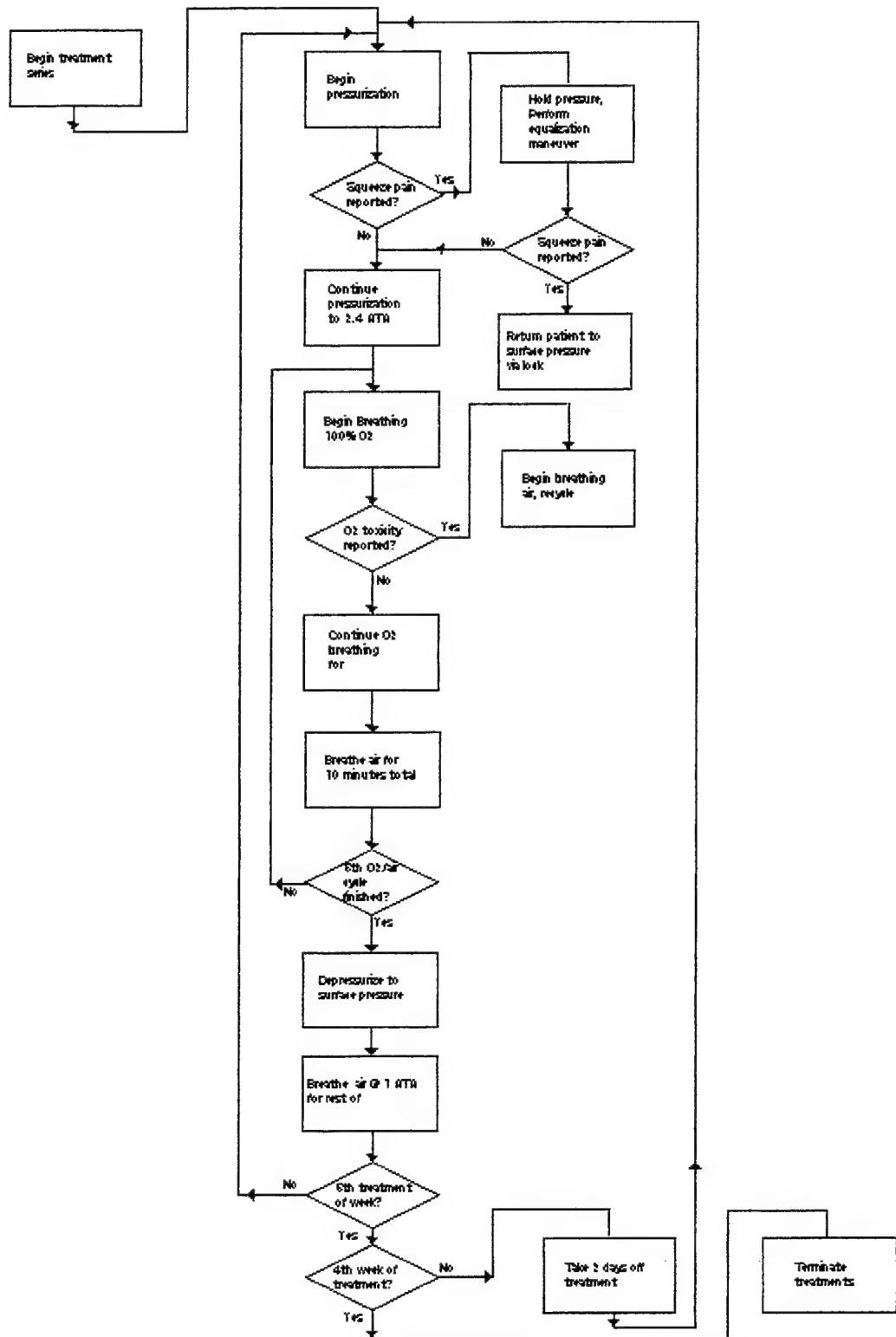


Figure 3: Flow diagram illustrating fundamental physical process steps for hyperbaric oxygenation treatment for wound healing.

Results

Composite Curing

During the term of this research project, this investigator reviewed the heuristics, code, tasks, and numerous rules and provided feedback to Dr. Abrams as she completed work on the several modules that constituted an automated discovery system for composite materials process control. Because envisionment building could be a very lengthy (and intrinsically speed limiting step) it was suggested that real-time process control should be separated from envisionment (to which it had been linked in the initial system design). This was done, as indicated in Figure 1 above, and the STUDY.DATA task could be executed in real-time with incoming process data. In its final form, the control module, known as the Process Experimental Theory Evaluator (PETE), combined the STUDY.DATA and RULE.BUILDER subtasks to spawn new heuristics to expand on the starting envisionment (model). Figure 4 shows a simple initial envisionment and a new version produced by the addition of a conjectured episode derived by PETE.

The completed system successfully produced a variety of envisionments of the composite curing process, and they showed autoclave control behaviors ranging from destructive through successful. By iteration, the system created an efficient control envisionment that produced successful cycles employing polymer cure exothermy, when exotherm heuristics had not been included in the original KB. Starting with a rudimentary envisionment, PETE was capable of creating envisionments that could be used for effective autoclave control within three or four of "discovery cycles". This was a very favorable outcome demonstrating functional discovery.

Other envisionments were produced that yielded characteristic control behaviors that resembled those of different human operators in ways suggesting "personality". For example, an apparently "conservative" envisionment achieved a successful cure cycle by repeatedly turning part heating on and off throughout the process, cycling around the ideal part temperature curve, yet expending more controller "effort". Distinct "dumb" envisionments produced burned part outcomes by not properly relating control measures to process indicators---indicating non-robust heuristics.

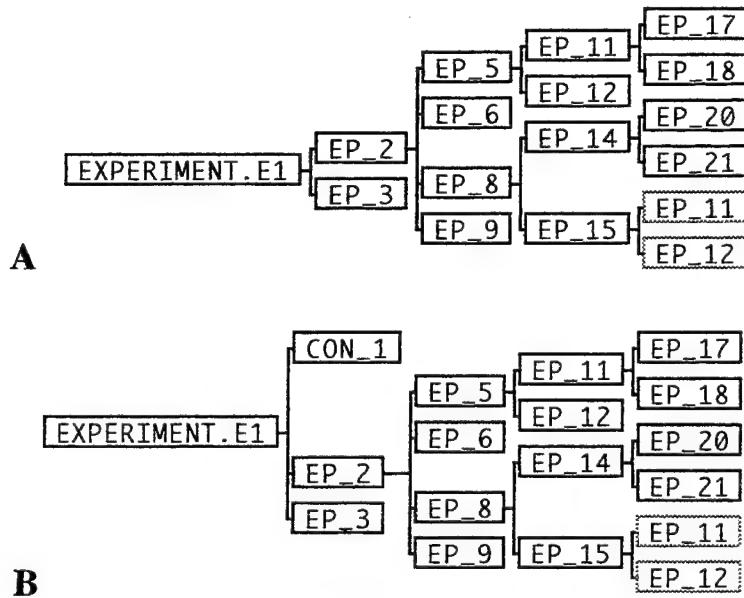


Figure 4: Browser graphic prints showing a simple initial composite curing envisionment (A) and subsequent refined envisionment (B) produced by PETE by adding a new conjectured process episode (state) encountered during control task.

Hyperbaric Oxygenation

The first step in the modeling process was to create the process rules describing the physical aspects of the wound healing chamber pressurization and gas cycles. The TSC browser graphic shown in Figure 5 depicts the linear sequence of episodes generated in an envisionment based solely on these HBO chamber physical process rules.



Figure 5: Browser graphic depiction of relationship of episodes in a TSC envisionment of the physical aspects of HBO chamber pressurizations and alternating periods of pure oxygen and air ventilation.

Perhaps the most substantial problem or set of problems was involved in generating the heuristics that portrayed cellular, tissue, and physiological processes involved in wound healing. This is a very complex business, given that healing involves activation of immune system cells, formation of granulation tissue ("scab"), ingrowth of new blood vessels, growth of new connective tissue, formation of new epithelial (skin surface) tissue, and contraction of the wound volume. Initial efforts focused on representing cellular responses to alternating cycles of oxygen and air breathing during an HBO treatment. It rapidly became apparent that working at this "low" (cellular) level of abstraction might elucidate some of the important tissue-level processes involved in enhanced healing, but would not provide much opportunity for describing and discovering processes important at the "patient level" of HBO. This awareness lead to a formulation of a general design strategy for TSC knowledge bases, one involving the partitioning of heuristics and envisionment/discovery tasks on the basis of levels of abstraction ("metalevels").

In this context, it was assumed that more effective modeling and discovery system functionality could be obtained if primary envisionment building concerned itself with the most "superficial" (or highest) level of reasoning about processes. Then, once these superficial processes were represented and assumptions were made, outcomes might be represented and explained in terms of deeper (or lower level) processes and relationships.

In terms of the HBO modeling problem, it would thus be most helpful to represent the gross physical aspects and patient responses to oxygenation treatments (i.e., measured cycle-related changes in the subject's oxygen levels and changes in wound volume). The most of the effort to build a fundamental KB for this project thus concerned itself with representing patient responses to changes in oxygen levels imposed by the physical HBO cycle. Several alternative responses were hypothesized at the outset, based on information contained in the HBO scientific literature [Davis and Hunt, 1988; Kloth et al., 1990]. First, the patient's measured transcutaneous O₂ partial pressure (TCPO₂) would faithfully reflect the "square wave" changes between air and 100% O₂, as imposed by the physical HBO cycle: 5 periods of 100% O₂ separated by briefer periods of air (~20% O₂) exposure. Second, there might be a "capacitively coupled square wave" response in which TCPO₂ levels would gradually approach a maximum, then fall to a uniform low during air breathing periods. Finally, there might be a "capacitively coupled square wave" response with a rising base level (as shown in Figure 6, below) where the TCPO₂ measurements showed a rising slope during air breathing periods.

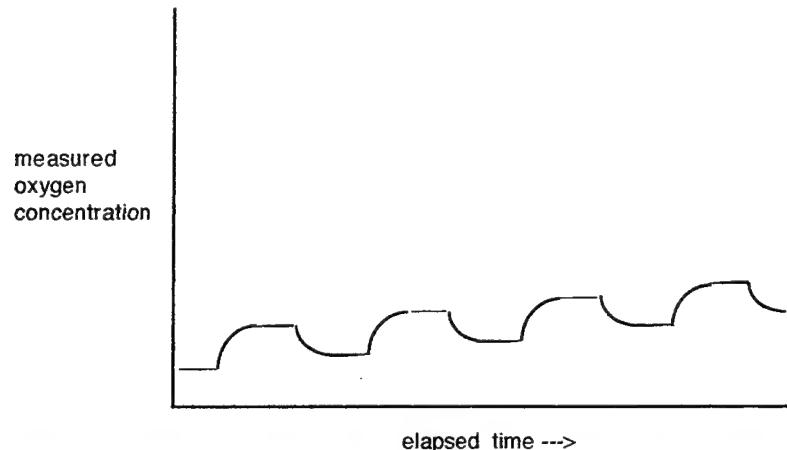


Figure 6: Hypothesized measured O₂ concentration levels during hyperbaric oxygenation treatment, based on the assumption that there is diffusion-limited penetration of oxygen into tissues and incomplete "washout" of high oxygen concentrations during air breathing periods. Four oxygen/air breathing cycles shown.

Given the lack of definitive information in the scientific literature demonstrating which model was correct, it was necessary to examine patient oxygenation data from HBO treatments. This required the configuration of a custom Macintosh-based data acquisition program (InfoScribe) to sample from the skin-attached gas electrodes most often used in hyperbaric medicine facilities. Data recorded from patients revealed that there was indeed a "capacitively coupled" rectangular wave pattern of tissue oxygen (Figure 7, below). Furthermore, in most cases, the baseline oxygen concentration achieved during air-breathing periods did *not* show a slow rise over the treatment interval, unlike that shown in Figure 6, above. This suggested that elevated oxygen concentrations rapidly "washed out" of the tissues nearly completely during air breathing, *unless there was a serious circulation (vascular flow) deficiency at the site(s) being measured*. These observations suggested important heuristics for the HBO KB for assessing patient responses (data) to treatment.

Perhaps the most important fundamental observation made about the continuous oximetry data was that the peak oxygenation levels for each oxygen-breathing period varied from cycle to cycle and between sites in some individuals. In "normal" or unaffected areas, electrodes recorded oxygen levels very close to the maximum possible based on the chamber pressure and oxygen concentration. In areas surrounding unhealing wounds, maximal oxygen tension recorded was frequently a small fraction of that measured at unaffected sites. Furthermore, at affected sites, these maximal oxygen levels could increase, decrease, or stay the same in successive periods of an HBO treatment session. Because the HBO scientific literature has reported that average peak oxygenation levels achieved at wound

sites may possibly predict the outcome of treatment [Davis and Hunt, 1988], we chose to focus on heuristics involving measured tissue oxygen values, as detailed below.

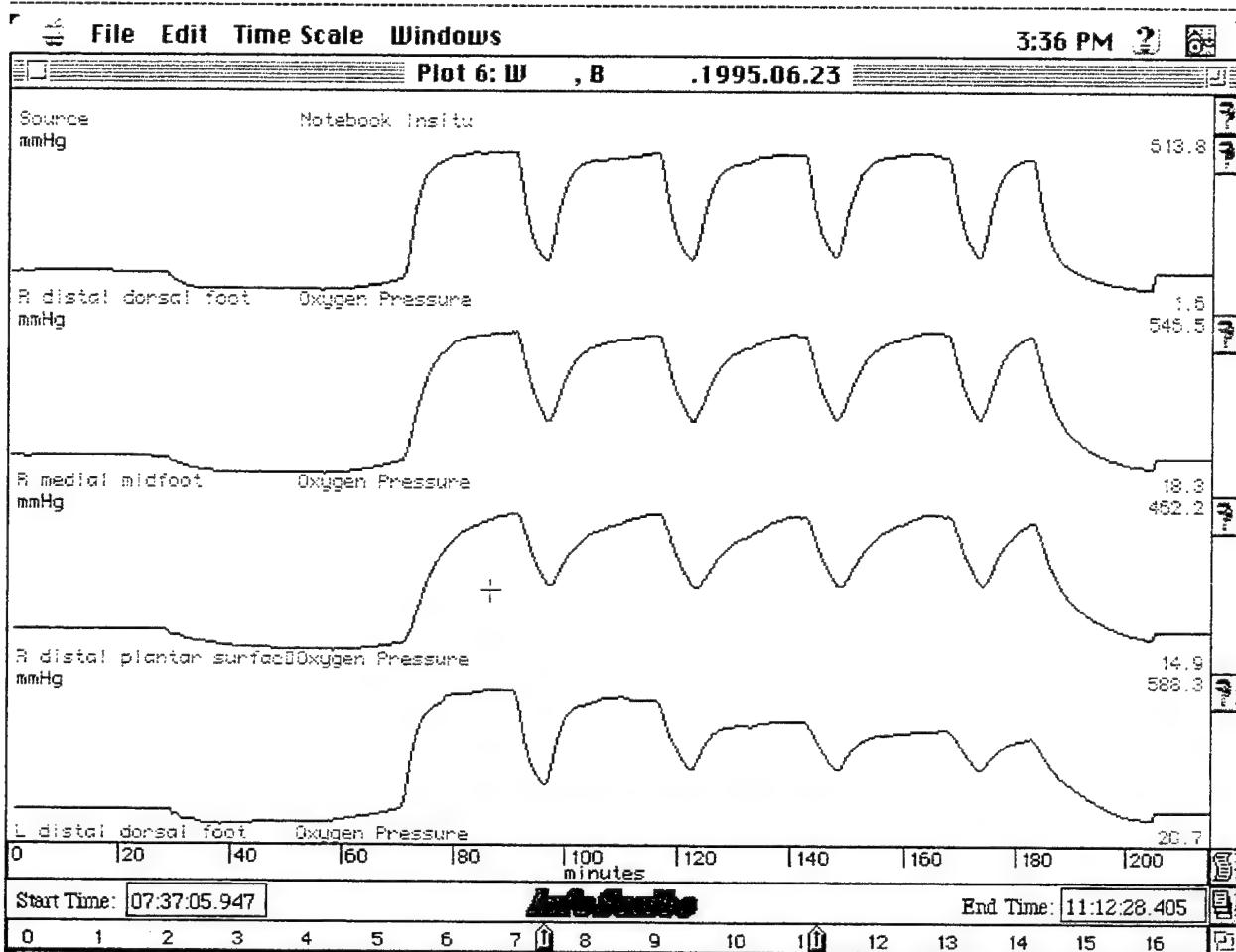


Figure 7: InfoScribe screen print showing measured TCPO₂ concentrations during HBO treatment in a patient with peripheral vascular disease. The lowest tracing shows decremental oxygenation.

Further study of the continuously recorded oximetry data revealed additional, previously undescribed features that might also be of use in assessing patient responses and in possibly predicting the long-term success or failure of HBO treatments for problem wound healing. Periodic-appearing variations in oxygen pressure were detected (Figure 8), falling in the same range (1-10 per minute) as described in the medical literature for "vasomotor waves" in arterial pressure [Schmidt et al., 1993; Seino et al., 1993]. Vasomotor waves have been shown to reflect oscillations in arterial smooth muscle tone (and hence, vessel caliber and flow) entrained and modulated by the sympathetic (autonomic) nerve fibers that innervate that muscle. Some medical research studies have reported that although the presence of these waves is a normal feature of arterial blood vessels, their magnitude may reflect disturbances in blood vessel control related to congestive heart failure and peripheral blood vessel (vascular) disease [Schmidt et al.,

1993; Seino et al., 1993; Hoffmann et al., 1994]. This latter condition is particular relevant to this project because HBO treatment is frequently used to heal ulcers and other wounds that are associated with the poor tissue oxygenation occurring with peripheral vascular disease.

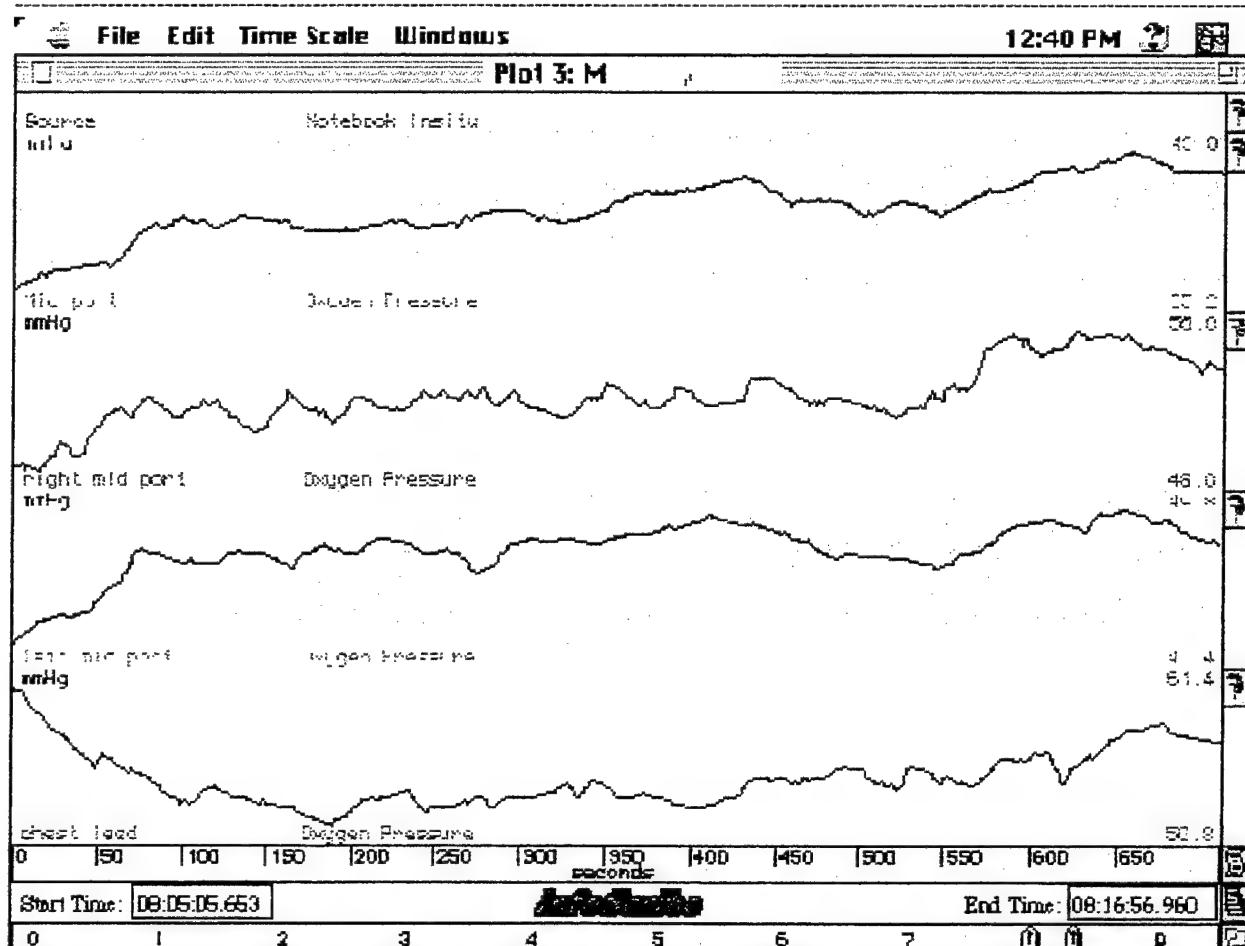


Figure 8: InfoScribe screen print showing low frequency oscillations in TCPO₂ during HBO treatment.

Neither slow wave oscillations nor variations in peak TCPO₂ values were noted in electrode test chambers subjected to the wound healing regimen of 2.4 ATA pressure and alternating air and 100 O₂ gas flows in the HBO pressure chamber. This provides confirmation that the waves and peak TCPO₂ variations observed were due to biological processes and *not* to electrode artifacts.

Given the variations in peak TCPO₂ levels observed in continuous recordings and published research reports that wound healing only occurred when mean oxygenated TCPO₂ exceed 3 times the basal values, modeling efforts focused on relating patient oxygen level responses to the physical cycles of HBO treatment sessions (as depicted in

Figure 5). Because published reports indicated that relatively large changes in TCPO₂ might be prognostic indicators for HBO treatment efficacy, "fuzzy" TCPO₂ level actors were created. Relative to maximal TCPO₂, very good (80-100%), good (60-79%), fair (40-59%), and poor (0-40%) were qualitative value actors assigned to encoded patient data. For example, on during the first oxygenation period, if the patient's TCPO₂ rose to 70% of the attainable maximum, the oxygenation level would be "good". The envisionment would thus show the possible patterns of peak TCPO₂ responses for the five oxygenation periods of an HBO treatment session. Figure 9 shows the directed process graph (browser print of episode relationships) for an relatively simple envisionment of an HBO treatment session, given the previously described fuzzy TCPO₂ levels and the pressurization chamber heuristics that produced the envisionment in Figure 5.

Execution speed for the most complex HBO envisionment was about 1800 episodes in 4 weeks on a 40 MHz 68040 Macintosh (Quadra 840AV) with 16 Mb RAM, and the envisionment halted short of execution because available working memory space was exceeded. Higher execution speeds (2700 episodes in 10 days) were obtained with the same envisionment on a 120 MHz Power PC 604 Power Macintosh with 32 Mb RAM.

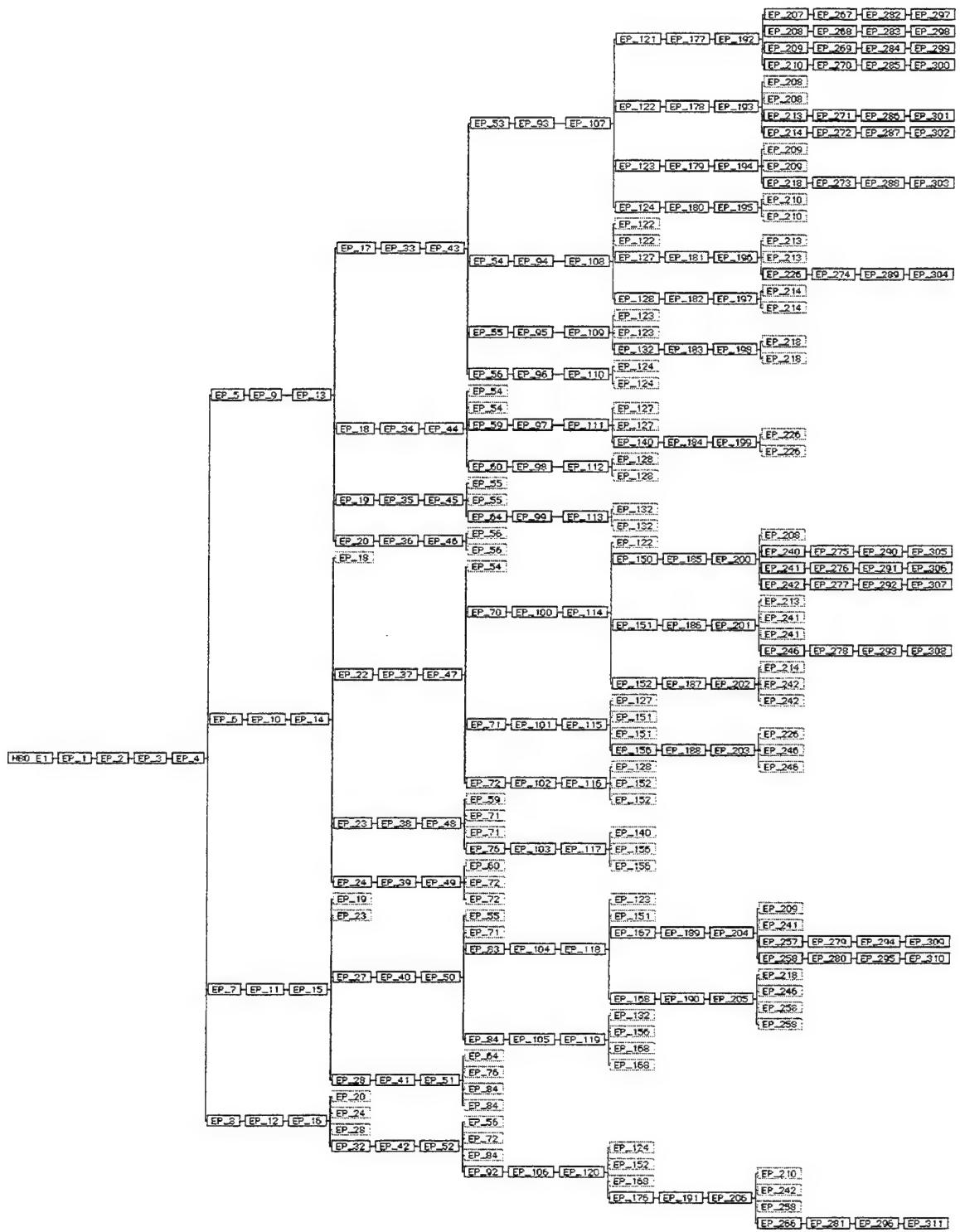


Figure 9: Browser graphic depiction of relationship of episodes in a envisionment of an HBO treatment session, illustrating the multiple possible combinations of patient "fuzzy" TCPO2 levels during separate oxygen-breathing periods.

Conclusions and Discussion

The autoclave/composite curing project demonstrated that a practical QP discovery and control system could be created using TSC. Starting with a relatively simple model, this discovery system was capable of learning new functional heuristics for composite curing based on process control interactions between a starting envisionment and a computer-based autoclave simulator. Partitioning of envisionment, rule-building, and control into separated tasks, with envisionments generated prior to cure/simulation cycles, allowed intelligent control tasks to execute reasonably in real time. This provides great promise for meeting critical fabrication and manufacturing needs, particularly in the aerospace industry. Beyond that, it suggests that practical systems can be created for automating and enhancing the processes involved in certain types of experimental laboratory research. Given the appropriate supporting equipment, it may be possible to construct computer-based scientific discovery systems capable of designing, conducting, and learning from their own experiments.

Not to diminish the long hours of work and clever programming performed by Dr. Frances Abrams in producing this system, the problem dealt with a relatively simple materials system, in terms of actual data variables and size of the knowledge base. In contrast, representing the biological systems involved in HBO and wound-healing was a far more complex task, because living systems adhere to very complicated, conditional process rules, and many subsystems involved in tissue repair are poorly characterized by available measurement techniques and not well understood. In fact, this complexity placed physical limits on execution speed, and the maximum working RAM space for a 16 Mb, 40 MHz 68040 Macintosh was easily exceeded during HBO envisionment building. To complete the work described here, it was necessary to obtain and configure a Power Macintosh (Motorola Power PC 604 RISC) running at 120 MHz with 32 Mb of RAM.

With four fuzzy range actors each assigned to possible patient levels of oxygenation and deoxygenation associated with HBO gas breathing cycles, the envisionments generated amounted to combinatorial explosions, with large numbers of episodes and possibly different patterns of responses to treatment. Initial data indicate that there are probably many fewer possible response patterns to be seen with HBO treatments of real patients. Further work is clearly necessary to "pare down" envisionments and to uncover as yet undescribed different patterns of response to treatment. Further work also remains to be done on data handling functions and tasks in order to "close the loop" in allowing full rule building and discovery to be implemented, as it was in the composite curing application.

Although oxygen (TCPO₂) measurements clearly represented the most readily acquired and appropriate data for assessing within-process patient responses to HBO, it was more difficult to obtain data that reliably measured overall treatment outcome. Given that objective measurements of tissue (e.g., blood vessel or scar) growth would be very difficult to obtain in a "non-invasive" (non-destructive) manner fro patients, it was originally proposed to use medical personnel as "virtual sensors" by encoding their subjective wound assessment data for each individual.

However, during the course of this project, a new method was demonstrated for using laser scanning to measure and to create precise volumetric models of wounds. Developed as a collaboration between the Hyperbaric Medicine facility of the 74th Medical Group (SGPH) and the Computer Anthropometry Laboratory (CARD Lab) of Armstrong Laboratory at Wright-Patterson AFB, the wound scanning system is capable of producing detailed, automatically segmentable volume/colorimetric models at the time of each HBO treatment without contact or interference with wounds or healing tissues. Preliminary assessment studies of wound healing with laser scanning are currently under way, and we propose to extend the present project by incorporating those wound scan data into a "closed loop" model of HBO-mediated wound healing.

Beyond incorporating wound measurement data, much additional work remains to be done to establish the prognostic significance of the different patterns of TCPO₂ changes noted in patient recordings. Furthermore, although CO₂ (TCPCO₂) and skin temperature data are also being recorded by the monitoring system, they have yet to be studied in correlation with progress of wound healing. Thermal measurements may be particularly relevant because skin temperature regulation (like oxygen delivery) is related to blood vessel perfusion in a region.

Because our initial observations seem to be novel, and we could find no account in the medical literature of the periodic-appearing waves observed in continuous TCPO₂ data recordings, the findings of our preliminary research are being reported in a paper being submitted for publication in a hyperbaric medicine journal.

References

Abrams, F. L. Process Discovery: Automated Process Development for the Control of Polymer Curing. Doctoral Dissertation, School of Engineering, University of Dayton, 1995.

Davis, J.C. and Hunt, T.K. Problem Wounds. The Role of Oxygen. Elsevier: New York, 1988.

De Kleer J. and Forbus, K. D. Building Problem Solvers. MIT Press, Cambridge, MA, 1993.

De Kleer, J. and Weld, D.S. Qualitative Physics: A Personal View, In:, D. S. Weld and J. De Kleer, Eds., Readings in Qualitative Reasoning about Physical Systems, pp. 1-8. Morgan Kaufmann, San Mateo, CA, 1992.

Forbus, K.D. Qualitative process theory. Artificial Intelligence 24:85-168, 1984.

Hoffmann U; Franzeck UK; Bollinger A. Low frequency oscillations of skin blood flux in peripheral arterial occlusive disease. Vasa 23(2):120-4,1994.

Hyperbaric Oxygen Committee. Hyperbaric Oxygen Therapy: A Committee Report. Undersea and Hyperbaric Medical Society: Bethesda, MD, 1992.

Kloth, L.C., McCulloch, J.M., and Feedar, J.A. (Editors). Wound Healing: Alternatives in Management. F.A. Davis Company: Philadelphia, 1990.

LeClair, S.R., Abrams, F.L., and Matejka, R.F. Qualitative process automation: Self-directed manufacture of composite materials. Artificial Intelligence for Engineering Design, Analysis and Manufacturing. 3(2):125-136, 1989.

LeClair, S.R., Abrams, F.L. Qualitative process automation. International Journal of Computer Integrated Manufacturing 2(4):205-211, 1989.

Oreskes, N., Shrader-Frechette, K., and Belitz, K. Verification, validation, and confirmation of numerical models in the earth sciences. Science 263:641-646, 1994.

Schmidt JA; Borgstrom P; Firestone GP; von Wichert P; Intaglietta M; Fronek A. Periodic hemodynamics (flow motion) in peripheral arterial occlusive disease. Journal of Vascular Surgery 18(2):207-15, 1993.

Seino Y; Tsukamoto H; Ohki K; Nakamura T; Kashiwagi M; Takano T; Hayakawa H. Abnormal cutaneous vasomotion and reduced cutaneous blood mass remain in congestive heart failure even with normalized cardiovascular hemodynamics. American Heart Journal 126(4):887-95, 1993.

Trelease, R.B. Developing qualitative process control discovery systems. Final Report, AFOSR Summer Faculty Research Program, 1994.

Uckun, S. Model-based reasoning in biomedicine. Critical Reviews in Biomedical Engineering 19:261-292, 1992.

Weld, D.S. and De Kleer, J., Eds. Readings in Qualitative Reasoning About Physical Systems. Morgan Kaufmann, San Mateo, CA, 1992.

DEVELOPMENT OF MASSIVELY PARALLEL EPIC HYDROCODE
IN CRAY T3D MASSIVELY PARALLEL COMPUTER

C.T. Tsai
Associate Professor
Department of Mechanical Engineering

Florida Atlantic University
777 Glades Road
Boca Raton, FL 33431

Final Report for:
Summer Faculty Research Program
Wright Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Wright Laboratory

February 1996

DEVELOPMENT OF MASSIVELY PARALLEL EPIC HYDROCODE
IN CRAY T3D MASSIVELY PARALLEL COMPUTER

C.T. Tsai
Associate Professor
Department of Mechanical Engineering
Florida Atlantic University

Abstract

The ELOOP subroutine in EPIC hydrocode have been converted to a CRAFT version of massively parallel subroutine in CRAY T3D computer. Several paralleled subroutines in ELOOP are then coupled with the sequential subroutines in EPIC using PVM to develop a coupled EPIC code. We gradually increase the number of paralleled subroutines implemented in the coupled EPIC code. The results show that the CPU time decreases as the number of parallel subroutines increase and the number of sequential subroutines decrease. When all the subroutines in the EPIC are paralleled, the coupled EPIC code will become a pure parallel code. Unfortunately, the slide interface algorithms are not suitable for parallel computing. Therefore, in order to completely parallel the EPIC code, a new parallel slide interface algorithm has to be developed.

DEVELOPMENT OF MASSIVELY PARALLEL EPIC HYDROCODE
IN CRAY T3D MASSIVELY PARALLEL COMPUTER

C.T. Tsai

1. INTRODUCTION

Parallel processing, the method of having many small tasks solve one large problem, has emerged as a key enabling technology in modern computing. The past several years have witnessed an increasing acceptance and adoption of parallel processing, for both high performance scientific computing and for more general purpose applications, was a result of the demand for higher performance, lower cost and sustained productivity. The acceptance of parallel processing has facilitated two major developments: massively parallel processors (MPPs) and the use of distributed computing[1].

A massively parallel processing (MPP) machine combine a few hundred to a few thousand CPUs in a single large cabinet connected to hundreds of GBytes of memory. MPPs offer enormous computational power and are considered to be one of the most powerful computers in the world.

Parallel processing is making tremendous impact on many areas of computer application. With the high raw computing power of parallel computers, it is now possible to address many applications that were until recently beyond the capability of conventional (sequential) computing techniques.

Many applications, such as weather prediction, biosphere modelling, pollution monitoring, finite element modelling, are modelled by imposing a grid over the domain being modelled. The entities within the grid elements are simulated with respect to the influence of other entities and their surroundings. In many cases, this requires a solution to large system of differential equations or solving a large sparse matrix algorithm. The granularity of the grid determines the accuracy of the model. Even for small number of grid points, a three dimensional coordinate system, and

a reasonable discretized time step, the modelling operation may involve trillions of operations. And so a moderate sized instances of these problems take an unacceptably long time to solve on serial computers[2].

The EPIC hydrocode is a lagrangian finite element code for higher velocity impact computations. It handles problems that contain large data set, exhaustive computations and has a need for faster and accurate results[3].

The Cray-T3D was chosen platform for converting the vectorized EPIC code to MPP code as it supports Multiple Instruction Multiple Data(MIMD) /Single Program Multiple Data (SPMD) model, CRAFT programming model and PVM programming model. Also it is closely coupled with the other Cray Parallel Vector Processor systems like Cray-YMP and Cray-C90 on which the EPIC code is already developed.

2. METHODOLOGY

The parallelization of the EPIC hydro code was carried out in the following steps:

1. Firstly, it was found that ELOOP (element computations) part of the EPIC hydro code consumed most of the cpu time. So, the ELOOP part was parallelized first.
2. Individual subroutines were parallelized using the MPP programming model and the CRAFT directives. The parallel subroutines were then compared with the vectorized subroutine.
3. Using PVM, the serial portion of the code was run on single PE (PE0) and the parallel subroutines were run on multiple PEs. PVM was used only for porting the parallelized subroutines into the EPIC code.

2.1. MPP PROGRAMMING MODEL

The MPP programming model[4] for the CRAY T3D system supports several styles of programming - data parallel, global address, work sharing and message passing. These styles may be individually used or combined in the same program. This model allows the user a range of control over the MPP hardware. This range extends from a low level control in which the programmer makes almost all of the decisions about how data and work are partitioned and distributed, to a high level of control where the programmer identifies where parallelism is located and lets the system determine best how to exploit it. The important elements of this

programming model are access and placement of data, parallel execution, local execution, worksharing, synchronization primitives, sequential I/O, subroutine interfaces and special intrinsic functions.

Data Parallel model

The MPP programming model distinguishes data objects into two categories :

- (1) private data (PE_PRIVATE), that are private to a task
- (2) shared data (SHARED), that are shared among all the tasks.

Private data objects reside on each PE, rather than spreading one copy over all of them. They are not accessible to any other task. The task that references a private object references its own private version of that object and therefore it is possible for private data objects associated with different PEs to have different values.

Shared data objects, on the other hand, are accessible to all tasks. They are not replicated and in case of arrays be distributed across multiple PEs.

In data parallel programming, data such as scalar or array are distributed over the memories of the PEs working on the program. In this programming model, the goal is to let as many PEs as possible perform on its own data (residing in its memory) rather than working on the data that is residing in another PE's memory.

In CRAFT, the data distribution is indicated by the compiler directives PE_PRIVATE and SHARED. Data that are not explicitly declared to be shared is, by default, private data. Variables and arrays can be explicitly declared as private with the PE_PRIVATE directive.

CDIR\$ PE_PRIVATE var1, var2 ... varn

All private data objects may be DATA initialized except those that occur in blank common, dummy arguments, automatic arrays, and those whose size is a function of N\$PES (number of tasks).

The shared data objects are declared with the SHARED directive.

CDIR\$ SHARED array1(dist1),array2(dist2) ...arrayn(distn)

The shared directive names the variables that are to be shared data objects and specifies the distribution across the PEs. Shared data object's distributions fall into two categories : shared scalars and dimensional distribution. Scalar variables are always allocated on a single PE, which

may differ for different variables. Dimensional distribution includes the following: Cyclic distribution, Generalized distribution, Block distribution and Degenerate distribution.

Cyclic distribution (:BLOCK(1)) assigns one element of shared array to each PE, returning to the first PE when every PE has an element. Generalized distribution (:BLOCK(n)) assigns blocks of 'n' elements of the array to successive PEs, where 'n' has to be an integer power of 2. The block distribution (:BLOCK) divides an array dimension into N\$PES blocks and allocates one block to each PE. The block size equals to array size divided by N\$PES. Degenerate distribution (:) forces an entire dimension to be allocated on a single PE.

Work Sharing

Executing the statements of program in parallel, and in the same PEs in which the data is distributed achieves higher performance for the Cray MPP system. Work sharing is achieved primarily by two ways: automatic arrays and shared DO loops.

Fortran array syntax or automatic arrays is one way to distribute work. A Fortran statement using array syntax and involving shared arrays encountered in a parallel region causes all processors to execute the statement. The compiler maximizes data locality i.e, the work is such distributed that tasks execute on its local data.

```
DIMENSION Z1(64), Z2(64), VNEW(64)
```

```
CDIR$ SHARED Z1(:BLOCK), Z2(:BLOCK), VNEW(:BLOCK)
```

```
.....
```

```
VNEW(I) = Z1(I) - Z2(I)
```

DOSHARED directive is the second way of achieving work sharing. As loops do not create parallelism, work sharing of DO loops is achieved by distributing iterations across all available tasks. Each task is assigned a set of iterations of a shared loop to execute. Shared loops do not guarantee the order in which iterations will be executed and lets the system execute iterations concurrently. There is an implicit barrier synchronization at the end of a shared loop. The example for DOSHARED directive from subroutine VOLUME is as follows:

```
CDIR$ DO SHARED (I) ON VNEW(I)
```

```
DO 10, I = 1, LNL1
```

```
VNEW(I) = Z1(I) - Z2(I)
```

10 CONTINUE

Private loops can be inside and outside the shared loop, but the shared loop must be tightly nested, the inner shared loop is executed as a private loop. The distribution mechanism for a shared loop affects program performance rather than correctness. Proper choice of iteration alignment provides higher degree of locality (when references in the iteration are close together). The aligned distribution mechanism is designed to place iterations within tasks on PEs where the references reside.

A private loop is executed only by the task that invokes it and no work is shared between tasks. Private loops define program behavior by defining the behavior of the individual tasks. Private loops have exactly the same semantics as loops in standard Fortran. No special syntax is required to specify a loop as private, as it is the default.

Shared to Private Coercion

The cardinal rule for distributed memory machines is to exploit data locality i.e, work with local data and avoid communication as much as possible. Performance without communication far exceeds that with communication.

There are two paradigms of CRAFT with no interprocessor communication. The highest performance is attained by shared to private coercion and the next paradigm is the PE_RESIDENT directive.

In shared to private coercion, an actual argument declared as shared array is passed to a corresponding dummy argument declared as a private array. This causes each PE to pass only its own data to the subroutine, which leads to the subroutine accessing array elements that are strictly local to the executing PE as private data without additional overhead.

The example illustrated here is from the STRAIN subroutine:

PROGRAM START_STRAIN

INTEGER L1,LN,M,LNL1

REAL Z1DOT (MXLB),Z2DOT (MXLB),HMIN (MXLB),EZDOT (MXLB),
2 EXDOT (MXLB),...

.....

CDIR\$ GEOMETRY GG(:BLOCK(1))

```
CDIR$ SHARED (GG) ::Z1DOT,Z2DOT,HMIN,EZDOT,EXDOT
```

```
.....
```

```
LNL1=(LN-L1+1)/N$PES
```

```
CALL STRAIN(L1,LN,EXDOT,EXYDOT,EYDOT,EZDOT,
```

```
* HMIN,Z1DOT,Z2DOT,.....LNL1)
```

```
END
```

```
SUBROUTINE STRAIN(L1,LN,EXDOT,EXYDOT,EYDOT,EZDOT,
```

```
* HMIN,Z1DOT,Z2DOT,.....LNL1)
```

```
C STRAIN computes strain rates
```

```
REAL Z1DOT(*),Z2DOT(*),HMIN(*),EXDOT(*),EZDOT(*)
```

```
.....
```

```
IF(IGEOM.EQ.1)THEN
```

```
do 10, i=1,LNL1
```

```
    EZDOT(I) = (Z1DOT(I)-Z2DOT(I))/HMIN(I)
```

```
    EXDOT(I) = 0.0
```

```
.....
```

```
10 CONTINUE
```

```
* enddo
```

```
.....
```

```
RETURN
```

```
END
```

In this example, the variables EXDOT, EZDOT, Z1DOT, Z2DOT and HMIN are defined as shared variables in the calling program, but are defined as private variables in the called subroutine STRAIN. This causes each PE to pass only its own data to the subroutine like the first element of these arrays are located in PE0, the second element of these arrays are located in PE1 and so on. So, when the DO loop is executed, all the executable statements work on local data. This, reduces the communication time and improves the performance of the parallel code.

3. PARALLIZATION OF INDIVIDUAL SUBROUTINES

All the subroutines discussed in the report were handled in a particular manner as discussed in subroutine VOLUME.

VOLUME

The VOLUME subroutine computes volumes, volumetric strains and strain rates. The VOLUME subroutine is called by the subroutine ELOOP. To simulate actual problem for the subroutine the variables and arrays are data initialized in the calling subroutine, in this case subroutine ELOOP. Among such variables and arrays are L1, LN ... X1(I) ...etc. As the subroutines in the EPIC program have common include files, the array size is made power of 2 for compiling in the Cray-T3D.

Firstly, the data in the subroutine are shared using different distributions like (:BLOCK(1)) and (:BLOCK). Work sharing is implemented explicitly by DOSHARED directives in the DO loop. A variable LNL1 is defined which is equal to LN-L1+1 and the original vectorized DO loop index J is eliminated

The Cray T3D being a dedicated machine, the real time clock function is used to measure the wall clock time. For better results, the number of times a subroutine is called is increased, so that the code accumulates some execution time.

Using BLOCK(1) and BLOCK data distribution and work sharing directives gave the following results as shown in Table 1. It is compared with the result obtained by running the code on Cray YMP. BLOCK data distribution gave better results than the BLOCK(1) as there is slightly less communication between the PEs in BLOCK than in BLOCK(1) data distribution. But in the final parallel code, BLOCK(1) distribution was used to take care for the case where the number of elements in the block is not a multiple of the number of PEs.

When run on Cray YMP, the time taken by the program is 0.763 secs. This is got by using the timex tool (as Cray YMP is not a dedicated machine and real time clock does not give the wall clock time).

Number of PEs	:BLOCK(1)	:BLOCK
1	2.794	2.543
2	1.121	1.112
4	0.923	0.883
8	0.475	0.462

Table 1: Comparison of :BLOCK and :BLOCK(1) data distribution directives

To further improve the performance of the subroutine the Shared to Private coercion technique was implemented. In this technique, shared arrays in the calling routine are passed to corresponding dummy arguments. Like x_1 is declared shared in ELOOP, but declared private in VOLUME. So, in the subroutine VOLUME each PE has x_1 in blocks of array-size divided by the number of PEs (array-size/n\$pes in this case LNL1/n\$pes). Due to this DO loops in the subroutine VOLUME also indexes upto LNL1/n\$pes, so that each PE work on its local data and no interprocessor communication takes place. The shared to private arrays are of arbitrary size and the local data of the subroutine are of arraysize/n\$pes size. Using shared to private coercion vastly imporoves the performance of the subroutine. The results of shared to private coercion implementation are shown in Table 2. It shows that the shared to private coercion code using 8 PEs is faster than YMP code which makes it cost effective. The basic rule for comparison of a parallel code on Cray-T3D and the vector code on Cray-YMP is that the parallel code running on 32 PEs has to be faster than vector code running on YMP for it to be cost effective.

Number of PEs	Wall clock time
YMP	0.763
1	2.298
2	0.900
4	0.513
8	0.281
16	0.183
32	0.136
64	0.118
128	0.108

Table 2: Wall clock timing of subroutine VOLUME

EGET

The EGET subroutine is used to initialize element variables from nodal variables. It is also called by subroutine ELOOP. The variables and arrays are data initialized in the calling subroutine.

To further improve performance, the shared to private coercion was implemented. As in the CRAFT program in the DOSHARED loop contains $J = L1 + I - 1$, shared to private coercion cannot be implemented as it is because it contains I on the right hand side of the assignment statement. So, a variable $II(I) = I$ was defined in the calling routine and shared to private coercion was implemented.

The performance gain by implementing the shared to private coercion principles can be seen in Table 3. Not only their is better performance compared to the corresponding CRAFT MPP code, but also is it more scalable.

Number of PEs	wall clock time
YMP	0.200
1	2.7615
8	0.338
16	0.185
32	0.109
64	0.0705
128	0.0519

Table 3: Wall clock timing of subroutine EGET

Similarly, the same parallel computing techniques i.e, data parallel, DO shared and Shared to private coercion were used in parallelizing the following subroutines. The Shared to Private coercion technique proved to be the best in most cases and the results are reported in Tables 4 and 5.

The results of certain subroutine are better than others. It can be attributed to Amdahl's law. The subroutines which has less sequential part in the code are better parallelized. In the following table, subroutine INCOAR is best parallelized as their is very less sequential part in it.

	GMCON	SOLID	INCOAR
Number of PEs	time	time	time
YMP	0.01820	1.078	1.078
8	0.01996	2.567	2.504
16	0.01160	1.440	1.050
32	0.00729	0.947	0.300
64	0.00545	0.746	0.155
128	0.00460	0.676	0.082

Table 4: Wall clock timing of subroutines GMCON, SOLID and INCOAR

SUBROUTINE	YMP	Number of PEs in T3D			
		4	8	16	32
FORCE	5.68	12.00	7.11	4.23	2.84
SOAK	1.10	1.93	1.15	0.83	0.62
STRAIN	0.14	0.32	0.19	0.12	0.08
CRUSH	5.71	15.59	8.04	5.48	3.21
HEBURN	1.19	3.94	2.19	1.35	0.77
IPRES	2.06	3.68	2.63	2.11	1.91
MEGRU	9.12	15.39	9.44	6.48	4.11
VISCOS	2.44	6.40	4.17	3.02	2.40

Table 5: Wall clock timing of other subroutines in ELOOP

4. Amdahl's Law

A common observation regarding parallel processing is that every algorithm has a sequential component that will eventually limit the speedup that can be achieved on a parallel computer. (Speedup is the ratio between execution time on a single processor and execution time on multiple processors.) This observation is often codified as Amdahl's law , which can be stated as follows: if the sequential component of an algorithm accounts for $1/s$ of the program's execution time, then the maximum possible speedup that can be achieved on a parallel computer is s . For example, if the sequential component is 5 percent, then the maximum speedup that can be achieved is 20.

In the early days of parallel computing, it was widely believed that this effect would limit the utility of parallel computing to a small number of specialized applications. However, practical experience shows that this inherently sequential way of thinking is of little relevance to real problems. To understand why, let us consider a noncomputing problem. Assume that 999 of 1000 workers on an expressway construction project are idle while a single worker completes a "sequential component" of the project. We would not view this as an inherent attribute of the

problem to be solved, but as a failure in management. For example, if the time required for a truck to pour concrete at a single point is a bottleneck, we could argue that the road should be under construction at several points simultaneously. Doing this would undoubtedly introduce some inefficiency---for example, some trucks would have to travel further to get to their point of work---but would allow the entire task to be finished more quickly. Similarly, it appears that almost all computational problems admit parallel solutions. The scalability of some solutions may be limited, but this is due to communication costs, idle time, or replicated computation rather than the existence of "sequential components."

Amdahl's law can be relevant when sequential programs are parallelized incrementally. In this approach to parallel software development, a sequential program is first profiled to identify computationally demanding components. These components are then adapted for parallel execution, one by one, until acceptable performance is achieved. Amdahl's law clearly applies in this situation, because the computational costs of the components that are not parallelized provide a lower bound on the execution time of the parallel program. Therefore, this "partial," or "incremental," parallelization strategy is generally effective only on small parallel computers. Amdahl's law can also be useful when analyzing the performance of data-parallel programs, in which some components may not be amenable to a data-parallel formulation.

5. PVM and porting of the parallel subroutines in the EPIC hydrocode

PVM (Parallel Virtual Machine) is a subroutine library that supports parallel programming through a technique known as message passing. This style of parallel programming is an explicit method in which the application specifically requests that data be sent from one task to another, or between groups of tasks. While PVM was developed to support parallel applications across a network of heterogeneous computer systems, message passing can also be used to program MPP systems.

There is an Cray MPP version of PVM[6] which can be used in two modes. In a stand alone mode it can be used to program an application on the Cray-T3D system, like the message passing libraries supplied for other MPP systems. In a distributed mode, it can be coupled with the network version of the Cray-T3D, so that it could communicate with processes running on the Cray-YMP system or any other system that run PVM. The MPP version uses the hardware

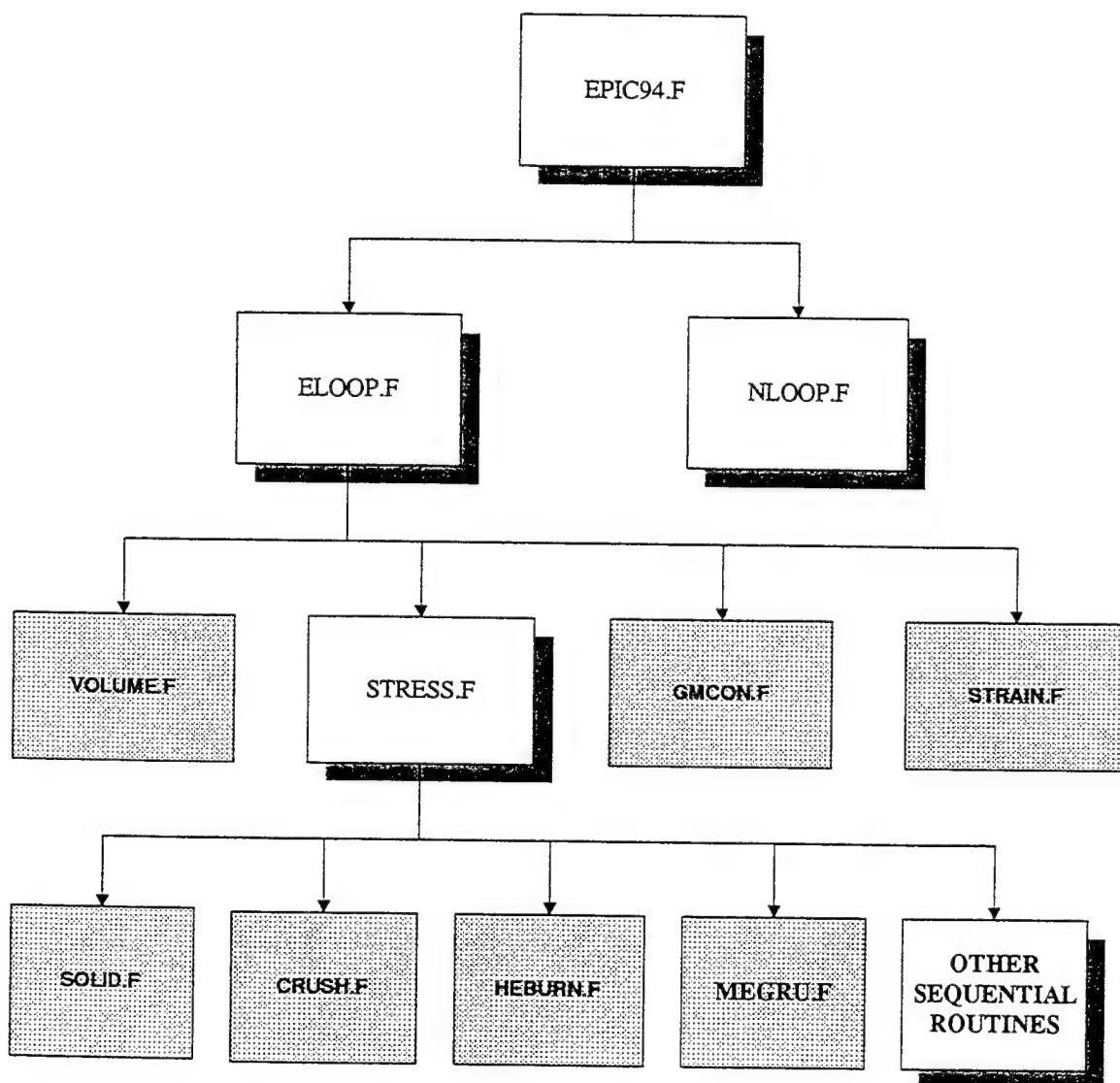
capabilities of the Cray-T3D system to handle communications between Cray-T3D processor elements, and uses TCP/IP and UDP data transfers to handle communications outside the Cray MPP system.

The EPIC PVM code can be divided into two parts: sequential part which is run on one PE usually PE0 of the MPP system and parallel part which is run on multiple PEs. As shown in the flowchart of the EPIC PVM code, subroutines called by ELOOP.F contains various shared subroutines whereas subroutines called by NLOOP contains all sequential subroutines. The parallelization and porting of parallel subroutines is a herietial process-subroutines in the lowest level (i.e., the shaded block in the flowchart) like SOLID.F, VOLUME.F, etc. are parallelized. Once all the subroutines contained in a calling subroutine (for example, STRESS.F in the flowchart) are shared, this calling subroutine is a completely parallel subroutine, otherwise the calling subroutine remains sequential. In this report, only 4 subroutines called by STRESS.F are parallelized so that STRESS.F is still a sequential subroutine. We are currently convering and porting the rest of the subroutines called by STRESS.F into parallel PVM code. Once it is completed, STRESS.F will become a parallel subroutine itself.

The EPIC PVM code starts in epic94.f which is the main program and is run on a single PE. It calls subroutines ELOOP which contains various parallel and sequential subroutines and NLOOP which contains all sequential subroutines.PVM is activated in the subroutine ELOOP when it calls a parallel subroutines. For example, PE0 passes a variable VOL_MSG to all active PEs just before calling parallel subroutine VOLUME. In subroutine VOLUME, data is copied to different PEs from PE0. Shared to Private coercion takes place in this subroutine when the shared subroutine VOLUME_SP3 calls private subroutine VOLUME_T3D. When subroutine ELOOP calls other sequential subroutines, PVM is not activated.

In the example in Appendix A, STRESS.F is the sequential part of the code. It calls various subroutines like SOLID and AVEP. Firstly, the code is run in PE0. When, SOLID_MSG is passed to the other PEs subroutine SOLID is called. Subroutine SOLID is part of SOLID_SP3.F. In this subroutine, the variables DVBAR, EDEV,etc are private to the PEs. Correspondingly, variables DVBAR, EDEV which are equal in value to DVBAR and EDEV respectively are

FLOWCHART FOR THE EPIC CODE



Sequential routines on PE0

Parallel routines on all PEs

shared. Then, this subroutine calls subroutine SOLID_T3D (part of SOLID_SP3.F) where shared to private coercion takes place. The time spent in message passing (Preroutines) is found and subtracted from the total time spent in the code as these message passing statements would be deleted once all the routines called by ELOOP is parallelized.

The same technique is used to port the other subroutines in ELOOP.

PORING RESULTS OF EXAMPLE PROBLEMS

The example problems of the EPIC manual were run and timed. The porting was done incrementally and so the results are presented in the same manner. Firstly, subroutine VOLUME was parallelized and ported which is case A. Case B corresponds to parallel subroutines VOLUME and SOLID. Case C corresponding to parallel subroutines VOLUME, SOLID and GMCON. Case D corresponding to parallel subroutines VOLUME, SOLID, GMCON, MEGRU, CRUSH and HEBURN. The results show that the wall clock time reduced as more subroutines were parallelized. Also, timing improved with the number of PEs. The results are shown in Table 6 and Table 7 for different example problems. They show that the CPU time in case D using 16 PEs are about twice of the YMP CPU time. This result is very impressive because only 4 parallel subroutines are ported. The CPU time will decrease as more parallel subroutines are ported into the EPIC code using PVM. We are currently porting more parallel subroutines into the EPIC code using PVM for the 1996 AFOSR summer research extension program.

YMP wall clock time is approximately 6 secs.

	4 PES	8 PES	16 PES
A	14.12	13.64	13.30
B	13.97	13.32	13.05
C	13.78	13.03	12.89
D	13.75	13.00	12.88

Table 6 : X1DAT (1-D Bar Impact)

YMP wall clock time is approximately 90 secs.

	4 PES	8 PES	16 PES
A	206.34	198.23	192.33
B	197.36	190.04	184.44
C	182.57	176.00	170.14
D	179.97	174.12	169.96

Table 7 : X2DAT (Cylinder Impact)

5. Conclusion and Futurework

The results of the EPIC subroutines showed that the wall clock time to run on 16 or 32 PEs on the T3D was less than the YMP wall clock time. This proves that the EPIC hydro code runs more cost effectively on the Cray-T3D than the Cray-YMP and other vector machines.

Also, as most of the arrays are single dimensioned, the code does not need explicit cache optimization. The Shared to Private coercion technique in the CRAFT model worked best for the EPIC code. Shared to Private coercion was better than both dataparallelism and DO shared directives. Not only, is Shared to Private a better algorithm in terms of speedup but also in terms of scalability.

For data parallelism, the :BLOCK was better than :BLOCK(1) scheme. But the later scheme was adopted for handling problems which will have node numbers in a block not multiples of 2.

The incremental approach of parallelizing individual subroutines and then porting them one by one by using PVM message passing techniques worked best for the EPIC code. The advantages of this method are ease in checking the correctness of the code and ease in finding the wall clock time.

Future work would involve porting all the subroutines called by ELOOP and increase the overall parallelism of the Parallel EPIC code. For some of the subroutines which are not easily parallelized using the above parallelizing techniques, new and improved parallel algorithms have to be developed. Finally, all the PVM statements have to be eliminated to get a fully parallelized

parallelized EPIC research code in CRAFT.

6. REFERENCES

- [1] Al Geist et al. PVM: A User's Guide and Tutorial for Networked Parallel Computing. The MIT Press, Cambridge, Massachusetts, 1994.
- [2] Vipin Kumar et al. Introduction to Parallel Computing: Design and Analysis of algorithms. The Benhamin/Cummings Publishing Company, Inc, 1994.
- [3] G.R. Johnson et al. User Instructions for the 1994 version of the EPIC Research code. Alliant Techsystems Inc, 1994.
- [4] Douglas M. Pase et al. MPP Fortran Programming Model. Cray Research Inc, 1993.
- [5] SR-2504 6.2. Cray MPP Fortran Reference Manual. Cray Research Inc, 1994.
- [6] SR-2501 3.0. PVM and HENCE Programmer's Manual, 1994.

APPENDIX A

STRESS.F

```
*$  
      SUBROUTINE STRESS(L1,LN,M,MT,MEOS,INCOMP,BI,BJ,BM,BP,CI,CJ,CM,CP,  
2           DI,DJ,DM,DP,DVBAR,DVDOT,EDOT,ENSUM,EXDOT,EXYDOT,  
3           EXZDOT,EYDOT,EYZDOT,EZDOT,HMIN,PBAR,Q,QMAX,SBAR,  
4           SMAX,SPINRZ,SS2,TSTAR,U,XSPIN,YSPIN,ZSPIN,DAM,  
5           DVOL,E BAR,EP,EPDOT,ES,ESPAR,E,I,CHECK,MBRIK,  
6           NODE1,NODE2,NODE3,NODE4,SX,SY,SZ,SXY,  
7           SXZ,SYZ,TSTART,VOL)
```

* Revised 93/08/18

C STRESS computes ELEMENT stresses

C

* Latest Revision FEBRUARY 1994

C called by ELOOP

* calls AVEP,CRUSH,HEBURN,IPRES,MEGRU,SOLID,UEOS,VISCOS

* Packaged in EPICS

C

* IMPLICIT NONE

INCLUDE 'fpvm3.h'

INTEGER SOLID_MSG

PARAMETER (SOLID_MSG = 1)

INTEGER GSIZ E

INTEGER IPVMST

INTEGER ISBUF

.....

IF(MT.EQ.1 .OR. MT.EQ.3 .OR. MT.EQ.5 .OR. MT.EQ.6)THEN

START_TIME = RTC()

IF (N\$PES .GT. 1) THEN

CALL PVMFINITSEND (PVMDATARAW,ISBUF)

IF (ISBUF .le. 0) then

WRITE (6,*) 'Error on init send '

endif

CALL PVMFBCAST (PVMALL,SOLID_MSG, IPVMST)

IF (IPVMST .NE. PVMOK) THEN

WRITE(6,*) 'ERROR ON BCAST SOLID ', IPVMST

ENDIF

ENDIF

CALL SOLID(L1,LN,INCOMP,DVBAR,EDEV,EDOT,EXDOT,EXYDOT,

2 EXZDOT,EYDOT,EYZDOT,EZDOT,PBAR,SBAR,SMAX,SPINRZ,

3 TSTAR,XSPIN,YSPIN,ZSPIN,DAM,E BAR,EP,EPDOT,ES,

4 ICHECK,M,NODE3(1),NODE4(1),SX,SY,SZ,SXY,SXZ,SYZ)

```

    ELAP_TIME = ELAP_TIME + RTC() - START_TIME
ENDIF
IF(MT.EQ.1)THEN
  IF(INCOMP.EQ.0)THEN
    IBRIK = MBRIK(1)
    IF(VFRACT.GT.0.0 .AND.
1      ((IGEOM.GE.4.AND.IGEOM.LE.7.AND.IBRIK.EQ.-2) .OR.
2      (IGEOM.EQ.8.AND.IBRIK.EQ.-6) )THEN
      CALL AVEP(L1,LN,LN,BI,BJ,BM,BP,CI,CJ,CM,CP,DI,DJ,DM,DP,
2        DVDOT,EDEV,EXDOT,EXYDOT,EXZDOT,EYDOT,EYZDOT,EZDOT,
3        HMIN,Q,SBAR,SS2,U,DVOL,EP,ES,ICHECK,M,MBRIK,NODE1,
4        NODE2,NODE3,NODE4,TSTART,VOL,SX,SY,SZ)
    ELSE
      .....
    RETURN
  END

```

SOLID_SP3.F

```

SUBROUTINE SOLID(L1M,LNM,INCOMPM,
2 DVARM,EDEVM,EDOTM,EXDOTM,EXYDOTM,
2 EXZDOTM,EYDOTM,EYZDOTM,EZDOTM,PBARM,SBARM,SMAXM,SPINRZM,
3 TSTAR,M,XSPINM,YSPINM,ZSPINM,DAMM,EBARM,EPM,EPDOTM,ESM,
4 ICHECKM,MM,NODE3M,NODE4M,SXM,SYM,SZM,SXYM,SXZM,SYZM)
IMPLICIT NONE
INTEGER MYPE,MY_PE
INTRINSIC MY_PE
INTEGER L1M,LNM,INCOMPM,MM,NODE3M,NODE4M
REAL time0, time1
INCLUDE 'VECTMX'
INCLUDE 'PREROUTINE'
REAL DVARM (MXLB),EDEVM (MXLB),EDOTM (MXLB),EXDOTM (MXLB),
2 EXYDOTM(MXLB),EXZDOTM(MXLB),EYDOTM(MXLB),EYZDOTM(MXLB),
3 EZDOTM (MXLB),PBARM (MXLB),SBARM (MXLB),SMAXM (MXLB),
4 SPINRZM(MXLB),TSTAR(MXLB),XSPINM (MXLB),YSPINM(MXLB),
5 ZSPINM (MXLB)
REAL DAMM (MXLB),EBARM(MXLB),EPM (MXLB),EPDOTM (MXLB),
2 ESM (MXLB),SXM (MXLB),SYM (MXLB),SZM (MXLB),
3 SXYM (MXLB),SXZM (MXLB),SYZM(MXLB)
INTEGER ICHECK(MXLB)
REAL DVBAR (MXLB),EDEV (MXLB),EDOT (MXLB),EXDOT (MXLB),
2 EXYDOT(MXLB),EXZDOT(MXLB),EYDOT (MXLB),EYZDOT(MXLB),
3 EZDOT (MXLB),PBAR (MXLB),SBAR (MXLB),SMAX (MXLB),
4 SPINRZ(MXLB),TSTAR (MXLB),XSPIN (MXLB),YSPIN (MXLB),
5 ZSPIN (MXLB)

```

```

REAL DAM (MXLB),EBAR(MXLB),EP (MXLB),EPDOT (MXLB),
2   ES (MXLB),SX (MXLB),SY (MXLB),SZ (MXLB),
3   SXY (MXLB),SXZ (MXLB),SYZ(MXLB)
INTEGER ICHECKM(MXLB)
CDIR$ GEOMETRY GZ(:BLOCK(1))
CDIR$ SHARED (GZ) :: DVBAR,EBAR,EDEV,EDOT,EPDOT,EP
CDIR$ SHARED (GZ) :: EXDOT,EYDOT,EZDOT,EXYDOT,EYZDOT,EXZDOT
CDIR$ SHARED (GZ) :: SBAR,SX,SY,SZ,SXY,SYZ,SXZ,SPINRZ
CDIR$ SHARED (GZ) :: XSPIN,YSPIN,ZSPIN,TSTAR,ES,SMAX,PBAR,DAM,ICHECK
COMMON /SP3_SOLID1/ DVBAR,EBAR,EDEV,EDOT,EPDOT,EP,
2   EXDOT,EYDOT,EZDOT,EXYDOT,EYZDOT,EXZDOT,
3   SBAR,SX,SY,SZ,SXY,SYZ,SXZ,SPINRZ,
4   XSPIN,YSPIN,ZSPIN,TSTAR,ES,SMAX,PBAR,DAM,ICHECK
* FOR LOCAL VARIABLES TO ENSURE THE END MASTER, COPY WORKS
INTEGER L1P,LNP,INCOMPP,MP,NODE3P,NODE4P
REAL C0P,C1P,C2P,C3P,C4P,C5P,C6P,C10P
REAL AMP,ANP
REAL DENP
REAL SPHP
REAL TEMP1P, TROOMP, TMELTP, TZEROP
INTEGER MTP,MDP
INTEGER IDAMP, IFAILP, MODELP
REAL PMINP, GP
COMMON /SP3_SOLID2/ L1P,LNP,LNL1,INCOMPP,MP,NODE3P,NODE4P,
2   C0P,C1P,C2P,C3P,C4P,C5P,C6P,C10P,
3   AMP,ANP,DENP,SPHP,TEMP1P, TROOMP, TMELTP, TZEROP,
4   MTP,MDP,IDAMP, IFAILP, MODELP,PMINP, GP
* SHOULD BE ONLY LOCAL NOT COPIED
INTEGER LNL1
INCLUDE 'MATERL'
INCLUDE 'MATERC'
INCLUDE 'MISC'
CDIR$ MASTER
C* Do a private to share coercion
DVBAR = DVBAR
EBAR = EBARM
EDEV = EDEVM
EDOT = EDOTM
EPDOT = EPDM
EP = EPM
EXDOT = EXDOTM
EYDOT = EYDOTM
EZDOT = EZDOTM
EXYDOT = EXYDOTM

```

```
EYZDOT = EYZDOTM  
EXZDOT = EXZDOTM  
SBAR = SBARM  
SX = SXM  
SY = SYM  
SZ = SZM  
SXY = SXYM  
SYZ = SYZM  
SXZ = SXZM  
SPINRZ = SPINRZM  
XSPIN = XSPINM  
YSPIN = YSPINM  
ZSPIN = ZSPINM  
TSTAR = TSTARM  
ES = ESM  
SMAX = SMAXM  
PBAR = PBARM  
DAM = DAMM  
ICHECK = ICHECKM
```

C private so we can pass copies to other PEs

```
L1P = L1M  
LNP = LNM  
INCOMPP = INCOMP  
MP = MM  
NODE3P = NODE3M  
NODE4P = NODE4M
```

C distribute the local values

```
C * MISC variables DT, EPSILON and IGEOM are copied  
CDIR$ END MASTER, COPY( L1P, LNP, INCOMPP, MP, NODE3P, NODE4P,  
CDIR$2 DT, EPSILON, IGEOM )
```

```
MYPE = MY_PE()  
LNL1=(LNP - L1P + 1)/n$pes  
if (mod((LNP - L1P + 1), N$PES) .gt. MYPE) then  
    LNL1 = LNL1 + 1  
endif
```

C All these are from private but ... for now copy

C C0-C40 are from MATERC

C Rest are from MATERL

```
MTP = MTYPE(MP)  
MDP = MODEL(MP)  
C0P = C0(MP)  
C1P = C1(MP)  
C2P = C2(MP)  
C3P = C3(MP)
```

```

C4P = C4(MP)
C5P = C5(MP)
C6P = C6(MP)
C10P = C10(MP)
AMP = AM(MP)
ANP = AN(MP)
DENP = DEN(MP)
SPHP = SPH(MP)
TEMP1P = TEMP1(MP)
TROOMP = TROOM(MP)
TMELTP = TMELT(MP)
TZEROP = TZERO(MP)
IDAMP = IDAM(MP)
IFAILP = IFAIL(MP)
MODEL P = MODEL(MP)
PMINP = PMIN(MP)
GP = G(MP)

CDIR$ BARRIER
C      time0=rtc()
      ELAP_TIME = ELAP_TIME + RTC() - START_TIME
      CALL SOLID_T3D(L1P,LNP,INCOMPP,DVBAR,EDEV,EDOT,EXDOT,EXYDOT,
      2   EXZDOT,EYDOT,EYZDOT,EZDOT,PBAR,SBAR,SMAX,SPINRZ,
      3   TSTAR,XSPIN,YSPIN,ZSPIN,DAM,E BAR,EP,EPDOT,ES,
      4   ICHECK,MP,NODE3P,NODE4P,SX,SY,SZ,SXY,SXZ,SYZ,
      5   MTP,MDP,C0P,C1P,C2P,C3P,C4P,C5P,C6P,
      6   C10P,AMP,ANP,DENP,SPHP,TEMP1P,TROOMP,
      7   TMELTP,TZEROP,IDAMP,IFAILP,MODEL P,PMINP,
      8   GP,l nl1)
      START_TIME = RTC()
*
      time1=rtc()
CDIR$ MASTER
C      WRITE(6,*) 'SOLID time =', (time1-time0)*6.667e-09,
C      1      ' LNL1 = ', LN - L1 + 1
CDIR$ END MASTER
C Now we send the results back to the parent
CDIR$ MASTER
      DVBAR M = DVBAR
      EBARM = EBAR
      EDEVM = EDEV
      EDOTM = EDOT
      EPDM = EP
      EXDOTM = EXDOT
      EYDOTM = EYDOT

```

```

EZDOTM = EZDOT
EXYDOTM = EXYDOT
EYZDOTM = EYZDOT
EXZDOTM = EXZDOT
SBARM = SBAR
SXM = SX
SYM = SY
SZM = SZ
SXYM = SXY
SYZM = SYZ
SXZM = SXZ
SPINRZM = SPINRZ
XSPINM = XSPIN
YSPINM = YSPIN
ZSPINM = ZSPIN
TSTARM = TSTAR
ESM = ES
SMAXM = SMAX
PBARM = PBAR
DAMM = DAM
ICHECKM = ICHECK
C * changed value in routine
INCOMPM = INCOMPP
CDIR$ END MASTER
    RETURN
END
*$
SUBROUTINE SOLID_T3D(L1,LN,INCOMP,DVBAR,EDEV,EDOT,EXDOT,EXYDOT,
2 EXZDOT,EYDOT,EYZDOT,EZDOT,PBAR,SBAR,SMAX,SPINRZ,
3 TSTAR,XSPIN,YSPIN,ZSPIN,DAM,EBAR,EP,EPDOT,ES,
4 ICHECK,M,NODE3,NODE4,SX,SY,SZ,SXY,SXZ,SYZ,
5 MT,MD,C0M,C1M,C2M,C3M,C4M,C5M,C6M,
6 C10M,AMM,ANM,DENM,SPHM,TEMP1M,TROO MM,
7 TMELTM,TZEROM,DA MM,IFAILM,MODELM,PMINM,
8 GM,lnl1)
*
C SOLID computes the deviator and shear stresses for solid materials
C
* Latest Revision FEBRUARY 1994
C called by STRESS
* calls USTRNG
* Packaged in EPIC5
C
* IMPLICIT NONE

```

```

INTEGER L1,LN,INCOMP,M,NODE3,NODE4
INTEGER IDAMM, IFAILM, MODELM
*      L1 = first element of block
*      LN = last element of block
*      M = the material number
*      INCOMP flags incompressible materials
INCLUDE 'VECTMX'

CDIR$ PE_PRIVATE DVBAR,EDEV,EDOT,EXDOT,EXYDOT,EXZDOT,EYDOT,EYZDOT
CDIR$ PE_PRIVATE EZDOT,PBAR,SBAR,SMAX,SPINRZ,TSTAR,XSPIN,YSPIN
CDIR$ PE_PRIVATE ZSPIN,DAM,EBAR,EP,EPDO,ES,SX,SY,SZ,SXY,SXZ,SYZ
CDIR$ PE_PRIVATE ICHECK
CDIR$ PE_PRIVATE L1, LN, LNL1, INCOMP,M,NODE3,NODE4
CDIR$ PE_PRIVATE MT, MD
CDIR$ PE_PRIVATE C0M, C1M, C2M, C3M, C4M, C5M, C6M, C10M
CDIR$ PE_PRIVATE AMM,ANM,DENM,SPHM,TEMP1M,IFAILM,MODELM,PMINM,GM

REAL DVBAR (MXLB/N$PES),EDEV (MXLB/N$PES),
1   EDOT (MXLB/N$PES),EXDOT (MXLB/N$PES),
2   EXYDOT(MXLB/N$PES),EXZDOT(MXLB/N$PES),
3   EYDOT (MXLB/N$PES),EYZDOT(MXLB/N$PES),
4   EZDOT (MXLB/N$PES),PBAR (MXLB/N$PES),
5   SBAR (MXLB/N$PES),SMAX (MXLB/N$PES),
6   SPINRZ(MXLB/N$PES),TSTAR (MXLB/N$PES),
7   XSPIN (MXLB/N$PES),YSPIN (MXLB/N$PES),
8   ZSPIN (MXLB/N$PES)
*      EDOT(I) = total strain rate
*      PBAR(I) = pressure from previous cycle
REAL DAM (MXLB/N$PES),EBAR (MXLB/N$PES),
1   EP (MXLB/N$PES),EPDOT (MXLB/N$PES),
2   ES (MXLB/N$PES),SX (MXLB/N$PES),
3   SY (MXLB/N$PES),SZ (MXLB/N$PES),
3   SXY (MXLB/N$PES),SXZ (MXLB/N$PES),SYZ(MXLB/N$PES)
*      EBAR(I) = plastic strain
INTEGER ICHECK(MXLB/N$PES)
INCLUDE 'MISC'

C
REAL AMV,ANV,C0V,C1V,C10V,C2D9,C2V,C3V,C4V,C5V,C6V,DENV,DT2,GDT,
2   GDTI,PMINV,T1V,T2V,T3V,T4V,TDT,TGDT,TGDTI,TMELTV,TZEROV
*      variables ending in V are temporaries for vector quantities
*      which are constant in the current element block
INTEGER I,J,MT,MD
*      I is an index into vector arrays
*      J is an index into element arrays
REAL DS1(MXLB/n$pes),DS2(MXLB/n$pes),DSX(MXLB/N$pes),

```



```

DS1(I) = SXZ1(I)*SPINRZ(I)*TDT
DS2(I) = (SX1(I)-SZ1(I))*SPINRZ(I)*DT
EDOT(I) = AMAX1(EDOT(I),0.0001)
enddo
ELSE
* here IGEOM.EQ.8
DO I=1,LNL1
  SBAR(I) = (SX(I)+SY(I)+SZ(I))*THIRD
  SX1(I) = SX(I) - SBAR(I)
  SY1(I) = SY(I) - SBAR(I)
  SZ1(I) = SZ(I) - SBAR(I)
  SXY1(I) = SXY(I)
  SXZ1(I) = SXZ(I)
  SYZ1(I) = SYZ(I)
  DSX(I) = (YSPIN(I)*SXZ1(I) - ZSPIN(I)*SXY1(I))*TDT
  DSY(I) = (ZSPIN(I)*SXY1(I) - XSPIN(I)*SYZ1(I))*TDT
  DSZ(I) = (XSPIN(I)*SYZ1(I) - YSPIN(I)*SXZ1(I))*TDT
  DSXY(I) = (ZSPIN(I)*(SX1(I)-SY1(I)) + YSPIN(I)*SYZ1(I)
  1      - XSPIN(I)*SXZ1(I))*DT
  DSXZ(I) = (YSPIN(I)*(SZ1(I)-SX1(I)) + XSPIN(I)*SXY1(I)
  1      - ZSPIN(I)*SYZ1(I))*DT
  DSYZ(I) = (XSPIN(I)*(SY1(I)-SZ1(I)) + ZSPIN(I)*SXZ1(I)
  1      - YSPIN(I)*SXY1(I))*DT
  EDOT(I) = AMAX1(EDOT(I),0.0001)
enddo
ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCC
C If elements have a solid material (MT.EQ.1)
C Redefine material constants to eliminate pointers
* If the Johnson-Cook model (MD.EQ.1) is used then
* If there is no thermal softening (AMV.EQ.0.0) then
* Do for all elements
* Compute homologous temperature, TSTAR
* Compute allowable stress, SMAX, without thermal softening
* Enddo
* Else there is thermal softening (AMV.GT.0.0)
* Do for all elements
* Compute homologous temperature, TSTAR
* Compute allowable stress, SMAX, with thermal softening
* Enddo
* Endif
* If there is pressure hardening (C4V.GT.0.0) then

```

```

*      Do for all elements
*          Update allowable stress, SMAX
*      Enddo
*  Endif
*  If there is a maximum limiting stress (C5V.GT.0.0) then
*      Do for all elements
*          Update allowable stress, SMAX
*      Enddo
*  Endif
* Elseif the modified Johnson-Cook model (MD.EQ.2) is used then
*     Similar to above for MD.EQ.1
* Elseif the Zerilli-Armstrong FCC model (MD.EQ.3) is used then
*     Do for all elements
*         Compute homologous temperature, TSTAR
*         Compute allowable stress, SMAX
*     Enddo
* Elseif the Zerilli-Armstrong BCC model (MD.EQ.4) is used then
*     Similar to above for MD.EQ.3
* Elseif a User Model (MD.EQ.10) is used then
*     Call USTRNG to determine allowable strength for user model
* Endif
* If there is damage softening (C10.GT.0) then
*     Reduce SMAX due to damage
* Endif

```

C (continued below)

C

CCCCCCCCCC
CCCCCCCCCCCC

C

IF(MT.EQ.1)THEN

C0V	=	C0M
C1V	=	C1M
C2V	=	C2M
C3V	=	C3M
C4V	=	C4M
C5V	=	C5M
C6V	=	C6M
AMV	=	AMM
ANV	=	ANM
DENV	=	DENM
SPHV	=	SPHM
TEMP1V	=	TEMP1M
TROOMV	=	TROO MM
TMELTV	=	TMELTM

```

TZEROV = TZEROM
T1V   = TEMP1V - TROOMV
T2V   = 1.0/(SPHV*DENV)
T3V   = 1.0/(TMELTV-TROOMV)
T4V   = TEMP1V - TZEROV
IF(MD.EQ.1)THEN
    IF(AMV.EQ.0.0)THEN
*CDIR$   VECTOR
*CVD$    VECTOR
        DO I=1,LNL1
            TSTAR(I) = (T1V + ES(I)*T2V) * T3V
            TSTAR(I) = AMAX1(TSTAR(I),0.0)
            TSTAR(I) = AMIN1(TSTAR(I),1.0)
            SMAX(I)  = (C1V + C2V*EBAR(I)**ANV)
            1          * (1.0 + C3V*ALOG(EDOT(I)))
*           TSTAR(I) = AMIN1(AMAX1((T1V+ES(I)*T2V)*T3V,0.0),1.0)
*           SMAX(I)  = (C1V + C2V*EBAR(I)**ANV)
*           SMAX(I)  = (C1V + C2V*exp(anv*aalog(ebar(i))))
*           1          * (1.0 + C3V*ALOG(EDOT(I)))
        enddo
    ELSE
*CDIR$   VECTOR
*CVD$    VECTOR
        DO I=1,LNL1
            TSTAR(I) = (T1V + ES(I)*T2V) * T3V
            TSTAR(I) = AMAX1(TSTAR(I),0.0)
            TSTAR(I) = AMIN1(TSTAR(I),1.0)
            SMAX(I)  = (C1V + C2V*EBAR(I)**ANV)
            1          * (1.0 + C3V*ALOG(EDOT(I)))
            2          * (1.0 - TSTAR(I)**AMV)
*           SMAX(I)  = (C1V + C2V*EBAR(I)**ANV)
*           SMAX(I)  = (C1V + C2V*exp(anv*aalog(ebar(i))))
*           1          * (1.0 + C3V*ALOG(EDOT(I)))
*           2          * (1.0 - exp(amv*aalog(tstar(i))))
*           2          * (1.0 - TSTAR(I)**AMV)
        enddo
    ENDIF
    IF(C4V.GT.0.0)THEN
        DO I=1,LNL1
            SMAX(I) = SMAX(I) + C4V*PBAR(I)
        enddo
    ENDIF
    IF(C5V.GT.0.0)THEN
        DO I=1,LNL1

```

```

        SMAX(I) = AMIN1(SMAX(I),C5V)
    enddo
ENDIF
ELSEIF(MD.EQ.2)THEN
    IF(AMV.EQ.0.0)THEN
*CDIR$      VECTOR
*CVD$       VECTOR
    DO I=1,LNL1
        TSTAR(I) = (T1V + ES(I)*T2V) * T3V
        TSTAR(I) = AMAX1(TSTAR(I),0.0)
        TSTAR(I) = AMIN1(TSTAR(I),1.0)
        SMAX(I) = (C1V + C2V*EBAR(I)**ANV)*(EDOT(I)**C3V)
*
*      SMAX(I) = (C1V + C2V*exp(anv*log(EBAR(I))))*
*      1           exp(c3v*log(EDOT(I)))
    enddo
ELSE
*CDIR$      VECTOR
*CVD$       VECTOR
    DO I=1,LNL1
        TSTAR(I) = (T1V + ES(I)*T2V) * T3V
        TSTAR(I) = AMAX1(TSTAR(I),0.0)
        TSTAR(I) = AMIN1(TSTAR(I),1.0)
        SMAX(I) = (C1V + C2V*EBAR(I)**ANV)*(EDOT(I)**C3V)
*
*      1           * (1.0 - TSTAR(I)**AMV)
*      SMAX(I) = (C1V + C2V*exp(anv*log(EBAR(I))))*
*      1           exp(c3v*log(EDOT(I)))*
*      2           (1.0 - exp(amv*log(tstar(I))))
    enddo
ENDIF
IF(C4V.GT.0.0)THEN
    DO I=1,LNL1
        SMAX(I) = SMAX(I) + C4V*PBAR(I)
    enddo
ENDIF
IF(C5V.GT.0.0)THEN
    DO I=1,LNL1
        SMAX(I) = AMIN1(SMAX(I),C5V)
    enddo
ENDIF
ELSEIF(MD.EQ.3)THEN
    DO I=1,LNL1
        TSTAR(I) = (T1V + ES(I)*T2V) * T3V
        TSTAR(I) = AMAX1(TSTAR(I),0.0)
        TSTAR(I) = AMIN1(TSTAR(I),1.0)

```

```

*      SMAX(I) = C0V + C2V*exp(anv*alog(EBAR(I)))*
*      SMAX(I) = C0V + C2V*(EBAR(I)**ANV)
1      *EXP((T4V+ES(I)*T2V)*(-C3V+C4V*ALOG(EDOT(I))))*
      enddo
ELSEIF(MD.EQ.4)THEN
DO I=1,LNL1
  TSTAR(I) = (T1V + ES(I)*T2V) * T3V
  TSTAR(I) = AMAX1(TSTAR(I),0.0)
  TSTAR(I) = AMIN1(TSTAR(I),1.0)
  SMAX(I) = C0V + C1V
1      *EXP((T4V+ES(I)*T2V)*(-C3V+C4V*ALOG(EDOT(I))))*
* 2      + C5V*exp(anv*alog(EBAR(I)))
2      + C5V*EBAR(I)**ANV
      enddo
ELSEIF(MD.EQ.10)THEN
C     CALL USTRNG(L1,LN,M,EDOT,EBAR,ES,SMAX,lnl1)
      write (6,*) 'CALL to USTRNG ignored in subroutine SOLID'
ENDIF
IF(IDAMM.EQ.1 .AND. IFAILM.EQ.1 .AND. C10M.GT.0.0)THEN
  C10V = C10M
  c11v=1.0/c10v
  DO I=1,LNL1
    IF(DAM(I).GT.(1.0-C10V))THEN
      SMAX(I) = SMAX(I)*(1.0-DAM(I))*c11v
    ENDIF
    enddo
  ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCC
C Elseif elements have a crushable material (MTYPE(M).EQ.3)
* If standard model then
C   Do for all elements (vectorized)
C     Compute allowable stress, SMAX(I)
C   Enddo
* If there is a limiting maximum stress, C5
*   Do for all elements
*     Update allowable stress, SMAX
*   Enddo
* Endif
* Elseif Holmquist-Johnson-Cook model then
C   Do for all elements (vectorized)
C     Compute allowable stress, SMAX(I)
C   Enddo

```

```

* If there is a limiting maximum stress, C5
* Do for all elements
* Update allowable stress, SMAX
* Enddo
* Endif
* Endif
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCC
*
ELSEIF(MT.EQ.3)THEN
  IF(MODELM.EQ.1)THEN
    C1V = C1M
    C3V = C3M
    C4V = C4M
    C5V = C5M
    IF(IDAMM.EQ.0 .OR. IFAILM.EQ.0)THEN
      *CDIR$   VECTOR
      *CVD$   VECTOR
      DO I=1,LNL1
        SMAX(I) = (C1V + C4V*PBAR(I))*(1.0 + C3V*ALOG(EDOT(I)))
      ENDDO
      ELSE
        DO I=1,LNL1
          SMAX(I) = (C1V*(1.0-DAM(I)) + C4V*PBAR(I))
2         *(1.0 + C3V*ALOG(EDOT(I)))
        enddo
      ENDIF
      IF(C5V.GT.0.0)THEN
        DO I=1,LNL1
          SMAX(I) = AMIN1(SMAX(I),C5V)
        enddo
      ENDIF
    ELSEIF(MODELM.EQ.2)THEN
      C1V = C1M
      C2V = C2M
      C3V = C3M
      C4V = C4M
      C5V = C5M
      ANV = ANM
      PMINV = PMINM
      pminvd=1.0/pminv
      c4vd=1.0/c4v
      amdx=1.0/amax1(pminv,epsilon)
    ENDIF
  ENDIF
ENDIF

```



```

SYZ(I) = SYZ1(I) + GDT*EYZDOT(I)
VMISES(I) = SQRT( 1.5*( SX(I)*SX(I) + SZ(I)*SZ(I)
2           + SY(I)*SY(I))
3           + 3.0*( SXZ(I)*SXZ(I) + SXY(I)*SXY(I)
4           + SYZ(I)*SYZ(I)))
VMISES(I) = AMAX1(VMISES(I), EPSLON)
SMAX(I) = AMIN1(SMAX(I), VMISES(I))
FACTOR(I) = SMAX(I)/VMISES(I)
SX(I) = SX(I)*FACTOR(I)
SZ(I) = SZ(I)*FACTOR(I)
SY(I) = SY(I)*FACTOR(I)
SXZ(I) = SXZ(I)*FACTOR(I)
SXY(I) = SXY(I)*FACTOR(I)
SYZ(I) = SYZ(I)*FACTOR(I)
enddo
ELSE
DO I=1,LNL1
  SX(I) = SX1(I) + TGDT*EXDOT(I) + DSX(I)
  SY(I) = SY1(I) + TGDT*EYDOT(I) + DSY(I)
  SZ(I) = SZ1(I) + TGDT*EZDOT(I) + DSZ(I)
  SXY(I) = SXY1(I) + GDT*EXYDOT(I) + DSXY(I)
  SXZ(I) = SXZ1(I) + GDT*EXZDOT(I) + DSXZ(I)
  SYZ(I) = SYZ1(I) + GDT*EYZDOT(I) + DSYZ(I)
  VMISES(I) = SQRT( 1.5*( SX(I)*SX(I) + SY(I)*SY(I)
2           + SZ(I)*SZ(I))
3           + 3.0*(SXY(I)*SXY(I) + SXZ(I)*SXZ(I)
4           + SYZ(I)*SYZ(I)))
  VMISES(I) = AMAX1(VMISES(I),EPSLON)
  SMAX(I) = AMIN1(SMAX(I),VMISES(I))
  FACTOR(I) = SMAX(I)/VMISES(I)
  SX(I) = SX(I)*FACTOR(I)
  SY(I) = SY(I)*FACTOR(I)
  SZ(I) = SZ(I)*FACTOR(I)
  SXY(I) = SXY(I)*FACTOR(I)
  SXZ(I) = SXZ(I)*FACTOR(I)
  SYZ(I) = SYZ(I)*FACTOR(I)
enddo
ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCC
C Assign constants
* If 1-D or 2-D then
C Do for all elements (vectorized)

```



```

    EPDOT(I) = SQRT(C2D9*((EXDOT(I)-EZDOT(I))*(EXDOT(I)-
1      EZDOT(I))+EXDOT(I)-EYDOT(I)*(EXDOT(I)-EYDOT(I))-
2      +(EZDOT(I)-EYDOT(I))*(EZDOT(I)-EYDOT(I))-
3          + 1.5*( EXZDOT(I)*EXZDOT(I)-
4          + EXYDOT(I)*EXYDOT(I)-
5          + EYZDOT(I)*EYZDOT(I)))))
    EBAR(I) = EBAR(I) + EPDOT(I)*DT
    EP(I)   = EP(I) + ( (SX(I)+SX1(I))*EXDOT(I)-
1          + (SZ(I)+SZ1(I))*EZDOT(I)-
2          + (SY(I)+SY1(I))*EYDOT(I)-
3          + (SXZ(I)+SXZ1(I))*EXZDOT(I)-
4          + (SXY(I)+SXY1(I))*EXYDOT(I)-
5          + (SYZ(I)+SYZ1(I))*EYZDOT(I))-
6          * (1.0+DVBAR(I))*DT2
    ENDIF
    enddo
    IF(IGEOM.EQ.4)THEN
        INCOMP = 1
    ENDIF
    IF(IGEOM.GE.5 .AND. NODE3.EQ.0)THEN
        INCOMP = 2
    ENDIF
    ELSE
        here 3-D
        DO I=1,LNL1
            EDEV(I) = ( (SX(I)+SX1(I))*EXDOT(I)-
1                + (SY(I)+SY1(I))*EYDOT(I)-
2                + (SZ(I)+SZ1(I))*EZDOT(I)-
3                + (SXY(I)+SXY1(I))*EXYDOT(I)-
4                + (SXZ(I)+SXZ1(I))*EXZDOT(I)-
5                + (SYZ(I)+SYZ1(I))*EYZDOT(I))-
6                * (1.0+DVBAR(I))*DT2
            EXDOT(I) = EXDOT(I) - (SX(I)-SX1(I)-DSX(I))*TGDTI
            EYDOT(I) = EYDOT(I) - (SY(I)-SY1(I)-DSY(I))*TGDTI
            EZDOT(I) = EZDOT(I) - (SZ(I)-SZ1(I)-DSZ(I))*TGDTI
            EXYDOT(I) = EXYDOT(I) - (SXY(I)-SXY1(I)-DSXY(I))*GDTI
            EXZDOT(I) = EXZDOT(I) - (SXZ(I)-SXZ1(I)-DSXZ(I))*GDTI
            EYZDOT(I) = EYZDOT(I) - (SYZ(I)-SYZ1(I)-DSYZ(I))*GDTI
            IF(ICHECK(I).GE.0)THEN
                EPDOT(I) = SQRT(C2D9*((EXDOT(I)-EYDOT(I))*(EXDOT(I)-
1                  EYDOT(I))+EXDOT(I)-EZDOT(I)*(EXDOT(I)-EZDOT(I))-
2                  +(EYDOT(I)-EZDOT(I))*(EYDOT(I)-EZDOT(I))-
3                      + 1.5*( EXYDOT(I)*EXYDOT(I)-
4                      + EXZDOT(I)*EXZDOT(I))

```

```

5           + EYZDOT(I)*EYZDOT(I))))
EBAR(I) = EBAR(I) + EPDOT(I)*DT
EP(I)   = EP(I) + ( (SX(I)+SX1(I))*EXDOT(I)
1           + (SY(I)+SY1(I))*EYDOT(I)
2           + (SZ(I)+SZ1(I))*EZDOT(I)
3           + (SXY(I)+SXY1(I))*EXYDOT(I)
4           + (SXZ(I)+SXZ1(I))*EXZDOT(I)
5           + (SYZ(I)+SYZ1(I))*EYZDOT(I)
6           * (1.0+DVBAR(I))*DT2
ENDIF
enddo
IF(NODE3.EQ.0)THEN
INCOMP = 3
ELSEIF(NODE4.EQ.0)THEN
INCOMP = 4
ENDIF
ENDIF
RETURN
END

```

CHARACTERIZATION OF THE MECHANICAL PROPERTIES
OF MATERIALS IN A BIAXIAL STRESS ENVIRONMENT

John C. Lewis
Research Assistant
Department of Chemical and Materials Engineering

University of Kentucky
177 Anderson Hall
Lexington, KY 40506

Final Report for:
Summer Research Extension Program

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Wright Laboratory
Eglin AFB, FL

December 1996

CHARACTERIZATION OF THE MECHANICAL PROPERTIES
OF MATERIALS IN A BIAXIAL STRESS ENVIRONMENT

John C. Lewis
Graduate Student
Department of Chemical and Materials Engineering
University of Kentucky

Abstract

This paper presents a theory for the yielding behavior of materials under combined tension-torsion using the von Mises yield and flow criteria and accounting for the effects of testing machine stiffness. Experimental results are given for tests performed on OFE copper and 6061-T6 aluminum. These results are compared to estimates made using the von Mises and Tresca yield criteria. It is concluded that neither theory may be totally satisfactory in modeling the yielding response of real materials under multiaxial loading.

CHARACTERIZATION OF THE MECHANICAL PROPERTIES OF MATERIALS IN A BIAXIAL STRESS ENVIRONMENT

Introduction

For many years, a great deal of interest has been focused on the mechanical response of materials to multiaxial states of stress. For tests conducted under uniaxial stress, the material response can be simply expressed by a single stress-strain curve. Many mechanical properties can then be determined from this curve. For tests conducted under multiaxial stress, an effective stress and effective strain must be defined if the mechanical response is to be expressed as a single stress-strain curve[1]. For an isotropic material, the most commonly used initial yield conditions are the von Mises and the Tresca criteria which can be used to derive expressions for effective stress and effective strain.

This paper focuses on the analysis of a thin-walled tube loaded in tension and torsion. This combined loading produces a biaxial state of stress where there are only two non-zero components of the stress tensor. Equations for effective stress and strain are derived using the von Mises yield and flow criterion and considering the effects of machine stiffness[2] for tests where the linear and rotary displacements are given by ramp functions. Also, an experimental yield surface is constructed from tests performed on OFE copper and 6061-T6 aluminum. Finally, the validity of the von Mises and Tresca yield criteria is briefly discussed.

Theory

The following is a theory which derives an equation for effective strain using the von Mises yield and flow criterion and considering the effects of machine stiffness for tension-torsion tests where the linear and rotary displacements are given by ramp functions[3]. Consider a thin-walled tube of axial gage length L , cross-sectional area A , and some average radius R . Let ΔL denote the axial displacement of the combined specimen-machine system and Φ be its angle of twist. During a combined tension-torsion test

$$\Delta L = \Delta L_S^E + \Delta L_S^P + \Delta L_M , \quad (1)$$

and

$$\Phi = \Phi_S^E + \Phi_S^P + \Phi_M . \quad (2)$$

where the subscripts S and M denote specimen and machine and the superscripts E and P denote elastic and plastic, respectively. In this analysis it is assumed the machine undergoes only elastic deformations. The elastic displacements can be written

$$\Delta L_S^E = PL/AE. \quad (3)$$

$$\Delta L_M = P/k_1 . \quad (4)$$

$$\Phi_S^E = TL/AGR^2. \quad (5)$$

and

$$\Phi_M = T/k_2 . \quad (6)$$

Here P is the axial load, T is the torque, E and G are the elastic and shear moduli, respectively, and k_1 and k_2 are measures of machine stiffness. Using these relations, equations 1 and 2 can be rewritten as

$$\Delta L = (PL/AE) (1 + AE/k_1 L) + \Delta L_S^P . \quad (7)$$

and

$$\Phi = (TL/AGR^2) (1 + AGR^2/k_2 L) + \Phi_S^P . \quad (8)$$

Let the axial and angular displacements be given by ramp functions such that $\Delta L = at$ and $\Phi = bt$. In the early stages of deformation (near $t=0$), $\Delta L_S^P = \Phi_S^P = 0$. So equations 7 and 8 can be equated to their ramp functions and solved for P and T as

$$P = \frac{AEat}{L(1 + AE/k_1 L)} = C_1 t \quad (9)$$

and

$$T = \frac{AGR^2 bt}{L(1 + AGR^2/k_2 L)} = C_2 t. \quad (10)$$

Here, C_1 and C_2 are the initial slopes of the load-time and torque-time curves, respectively.

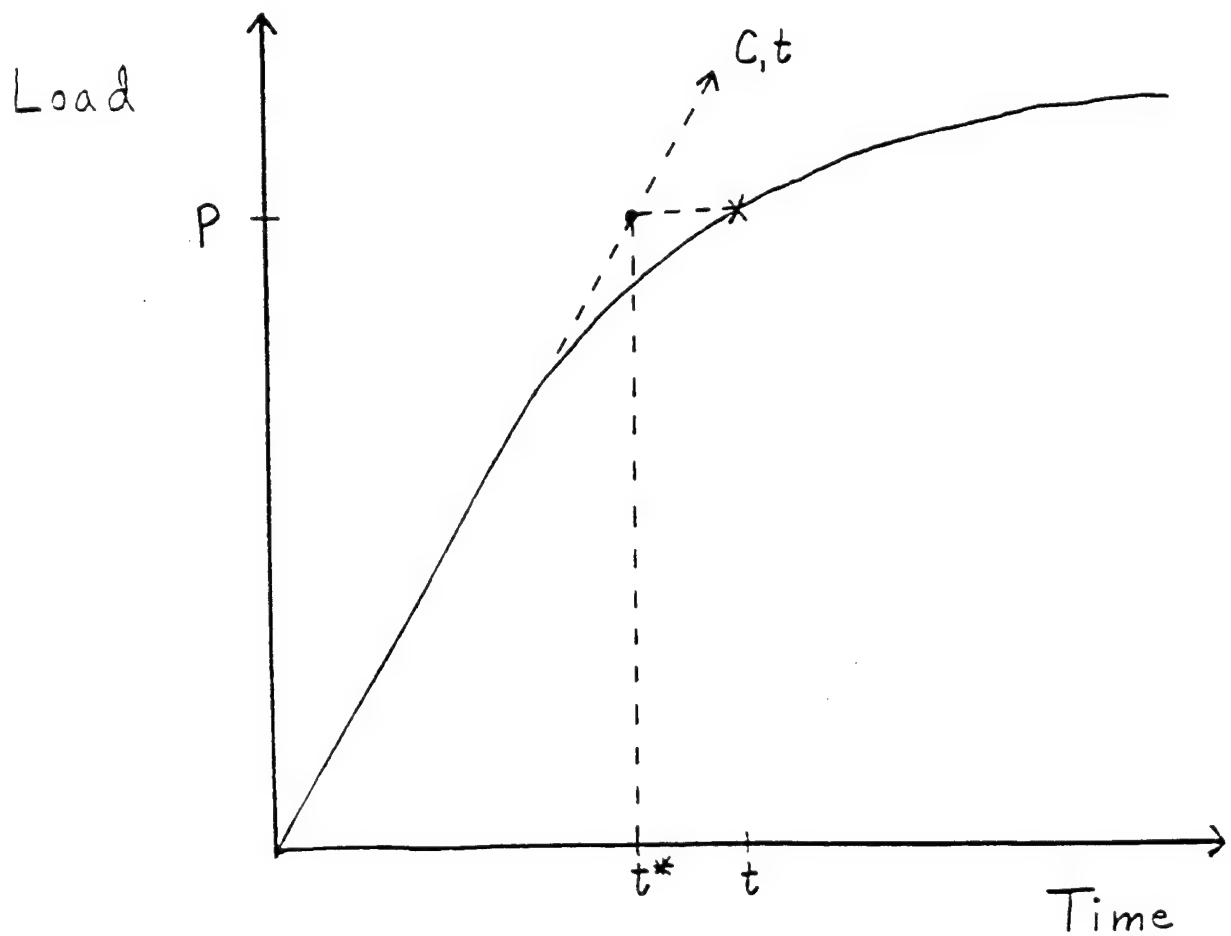


FIGURE 1. A typical load-time curve.

After the specimen begins to undergo plastic deformation equations 9 and 10 will no longer represent the load-time and torque-time curves. To determine the amount of plastic deformation the specimen has undergone at some time t , consider Fig. 1. Here t^* represents the time that would be required to reach load P if there were no plastic deformation and is defined by

$$t^* = P/C_1 . \quad (11)$$

Then the amount of plastic deformation in the specimen can be written

$$\Delta L^P = a(t - t^*). \quad (12)$$

The same idea can be applied to the torque-time curve. Thus the following equations are obtained:

$$\Delta L^P = a(t - P/C_1), \quad (13)$$

and

$$\Phi^P = b(t - T/C_2). \quad (14)$$

The corresponding plastic strains can be written

$$\varepsilon_z^P = \Delta L^P/L = (a/L)(t - P/C_1) \quad (15)$$

and

$$\gamma_{\theta z}^P = (R/L) \Phi^P = (bR/L)(t - T/C_2). \quad (16)$$

In a uniaxial test, the yield point can be defined by the stress at which the specimen has undergone some set amount of plastic strain. In a multiaxial test, the effective strain can be used in the same manner. For an isotropic material that obeys the von Mises flow criteria the effective plastic strain ε_P is defined through

$$d\varepsilon_P^2 = d\varepsilon_z^P{}^2 + 1/3 d\gamma_{\theta z}^P{}^2. \quad (17)$$

for tension-torsion, and for proportional strain paths

$$\varepsilon_P^2 = \varepsilon_z^P{}^2 + 1/3 \gamma_{\theta z}^P{}^2. \quad (18)$$

Now equations 15 and 16 can be substituted into equation 18 to obtain the following equation for effective plastic strain:

$$\epsilon_p^2 = [a/L (t - P/C_1)]^2 + 1/3 [bR/L (t - T/C_2)]^2.$$

(19)

By analogy with the offset yield stress often used in uniaxial tension tests, the multiaxial yield point can be defined as the point when ϵ_p reaches some set value.

Experimental

The specimens were thin-walled tubes of OFE copper and 6061-T6 aluminum, machined from round rod stock to within close tolerances. A sketch of the specimens is given in Fig. 2. The tests were performed by deforming the specimen under constant longitudinal displacement rate and a constant rate of change of the angle of twist. The effective strain rates were in the order of 0.001/s and no pre-stress was applied. The tubes were tested on an Instron model 1323 combined tension-torsion testing machine. The machine was programmed with a digital function generator and a desktop computer was used for data analysis and display.

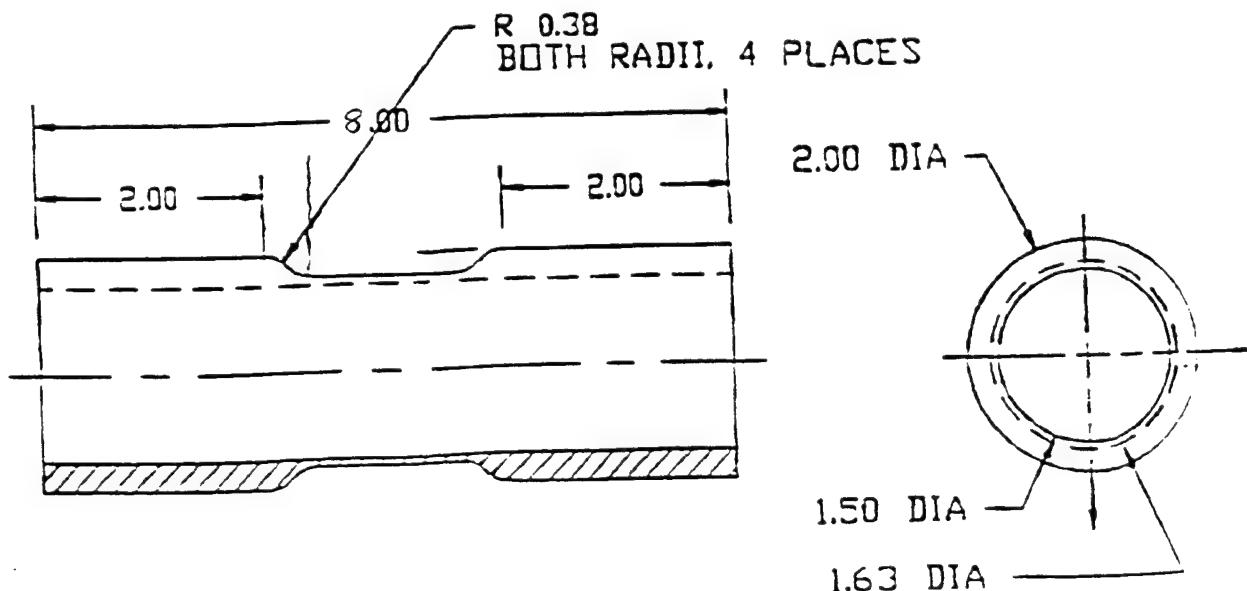


FIGURE 2. A sketch of the biaxial specimen. The units for the dimensions are inches.

Results and Discussion

For each test, the machine output was in the form of axial load and torque versus time. A yield point was determined from each curve using a 0.2% effective plastic strain. The results were plotted as the solid circles in Fig. 3 and 4. In each of these figures the solid curve represents the von Mises yield surface and the broken curve represents the Tresca yield surface.

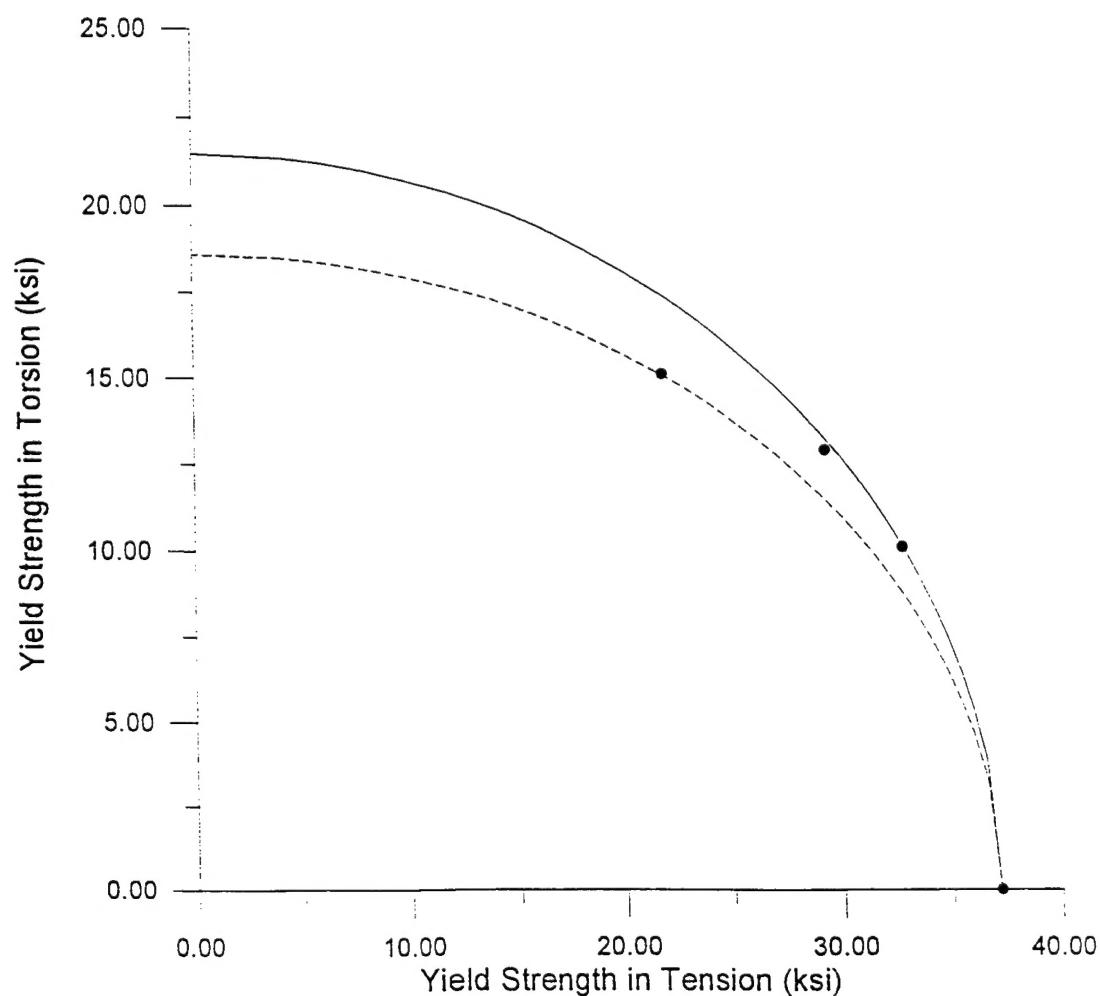


FIGURE 3. Experimental results for OFE copper. The solid curve represents the von Mises yield surface and the broken curve represents the Tresca yield surface.

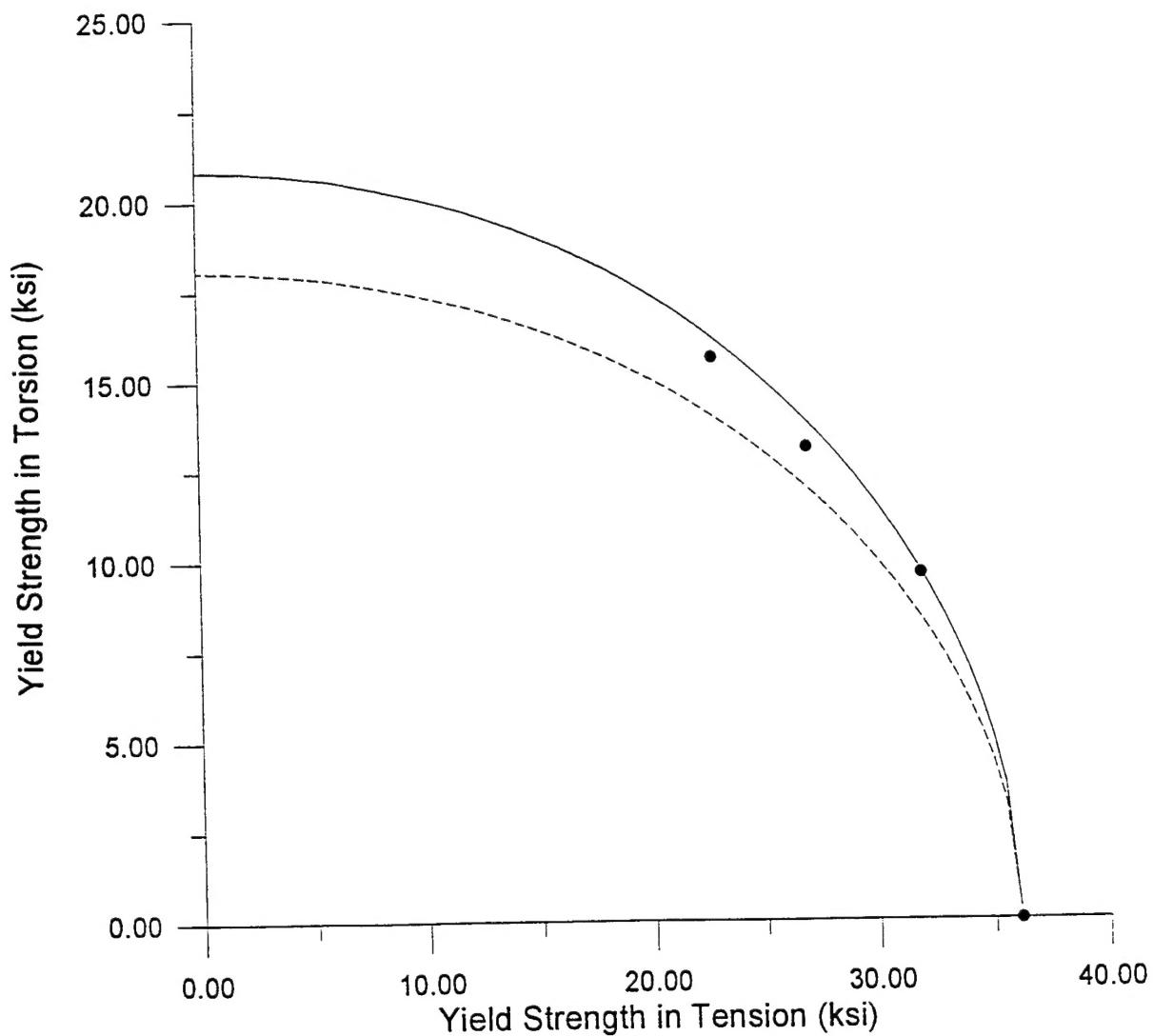


FIGURE 4. Experimental results for aluminum 6061-T6. The solid curve represents the von Mises yield surface and the broken curve represents the Tresca yield surface

Conclusion

Combined tension-torsion tests were performed on thin-walled tubes of OFE copper and 6061-T6 aluminum. A yield strength in tension and a yield strength in torsion were calculated for each test. These results were plotted along with yield surfaces generated using the von Mises and the Tresca yield criteria. It seems that neither of these yield criteria are totally satisfactory for predicting the yielding response of these real materials. It is noted, however, that all of the data points lie on or between the two curves representing these yield surfaces.

References

1. Stout, M.G., S.S. Hecker, and R. Bourcier, "An evaluation of anisotropic stress-strain criteria for the biaxial yield and flow of 2024 aluminum tubes," *J. of Engr. Matl. Tech.* (trans. ASME) **105** 242 (1983).
2. Hockett, John E., and Peter P. Gillis, "Mechanical testing machine stiffness: part I - theory and calculations," *Int. J. Mech. Sci.* **13** 251 (1971).
3. Lewis, J.C., "A theory for the testing of materials under combined tension-torsion," Report to AFOSR RDL Summer Research Program Office (1994).